LABORATOIRE D'INFORMATIQUE THEORIQUE
& APPLICATIONS DE MARSEILLE
L.I.T.A.M.
Faculté des sciences Economiques

UNIVERSITE d'AIX-MARSEILLE II

ISSN 0291 - 5413

INFORMATIQUE FONDAMENTALE & APPLICATIONS

Dépositaire
B.U. Sc. Eco.
Aix-Mars. II

SOMMAIRE

JUIN 1989

Adresse postale : FACULTé DES SCIENCES ECONOMIQUES
LITAM
14 rue Puvis de CHAVANNES 13001 MARSEILLE
91 90 13 20 P 420 et 421

EDITORIAL

DEFERLEMENT

E. Bianco

Le déferlement est cet étrange processus qui s'alimente d'énergies refoulées, accumulées, libérées brutalement sur un signal. Le processus dit "à relaxation", qui n'a rien à voir avec le transat sur la plage, est une suite de déferlements qui s'enchaînent en émettant, chacun, le signal de déclenchement du suivant.

Ainsi va notre pédagogie.

L'époque n'est pas si lointaine où l'information circulait à dos d'homme, si j'ose dire, au mieux par le téléphone arabe. La nouvelle, quand elle arrivait avait le temps d'être remâchée, disséquée, appréciée. Peut-être manquait-on un peu d'ouvertures de vues, il faut bien le dire; les strates de la société ne se connaissaient que par ouï-dire; beaucoup de légendes, de croyances, de rumeurs circulaient. En peu d'années ce mince filet d'eau rare est devenu un immense fleuve qui ne cesse de gonfler et de nous submerger. Avant, les gens communicaient; ils avaient peu de choses à se dire, ils s'en délectaient, ils jouaient avec le langage. Maintenant on n'a plus le temps, et de toute manière on a déja tout entendu, alors on condense sa pensée dans des concepts puissants et racés. Vive la langue de bois.

Toujours à l'image de nos fleuves, ces flots d'information charrient tous les immondices, tous les déchets, toutes les pollutions de la "civilisation". Mais, bien entendu, on y trouve aussi d'excellentes choses; tout-à-fait symétriquement il faut bien observer que les anciennes sources n'étaient pas exclusivement toutes d'essence pure.

Notre enseignement est à l'image de ce phénomène, les durées de prêche se sont accrues dans le rapport de un à six. Et pour faire bonne mesure, on a tendance à imiter les enseignements d'école d'ingénieurs, prolongeant de la sorte la tradition napoléonienne.

Quelles peuvent en être les conséquences ? Il me paraît que la simple observation de notre société peut apporter bien des réponses. On privilégie une certaine efficacité instantanée par rapport à un travail en profondeur qui exige une réflexion prolongée, donc, une apparente inactivité qui ressemble fort au sommeil. Donc il faut contrôler le travail. Bien entendu le travail des autres. Quel autre moyen, au moins dans le domaine de la "pensée", que d'exiger des traces écrites. Beaucoup de traces écrites. De plus en plus de traces écrites. Tout aboutit désormais à la constitution d'énormes dossiers que plus personne

ne lit d'ailleurs, mais cela permet de constituer à chacun une conscience tranquille. Voire.

Personne n'est véritablement tout-à-fait dupe, mais il ne s'agit que de s'installer dans un jeu abstrait. On s'entoure de dossiers comme les anciens seigneurs s'entouraient de remparts, plus les dossiers sont épais plus il sera facile de résister aux attaques. Car on en est ainsi à se livrer une véritable guerre de féodalité à féodalité, d'école à école. Dans un domaine où seule l'action commune pourrait être payante pour la communauté, on privilégie le "secret industriel" aussitôt complété par la mise en œuvre de moyens puissants pour pénétrer et pirater cet autre secret, volontiers de polichinelle qui est celui de l'adversaire. Que voila une guerre fort coûteuse dont il est absolument évident qu'elle ne peut aller que dans le sens de l'intérêt général et surtout pas favoriser l'accroissement de pouvoir local.

Des logiciels sont vendus bardés de complications pour retarder la pénétration de leur ridicule petit secret, ce qui en multiplie le prix par un facteur lamentable.

Les seigneurs qui se retranchaient depuis des siècles derrière des citadelles réputées imprenables n'avaient pas compris que le pouvoir se trouvait désormais ailleurs, par exemple dans les cours brillantes constellées de belles dames et tellement plus attractives pour les papillons. Notre industrie, jalouse de ses beaux secrets bien gardés et quelque peu fossilisée s'est largement fait déborder par l'industrie japonaise, qui construit peut-être plus fruste mais d'une manière tellement plus dynamique.

Dans l'enseignement et la recherche la situation est du même ordre. Aggravée par une tendance au gavage. On s'aligne au plus bas niveau. Que veut l'industriel ou l'entrepreneur ? s'interroge-t-on. La réponse tient en quelques phrases de langue de bois dont la sémantique se résume à peu près à ceci: de bons esclaves pas trop exigeants sur les salaires.

Mais la "formation de haut niveau" ?

Amusant. Quand on constate le niveau scientifique dont on a besoin réellement pour fabriquer les gadgets à gaver le consommateur, on prend conscience que l'embaucheur ait plutôt peur de mesurer son ignorance au regard de ce que paraîssent savoir les jeunes diplômés. d'autant que, se dit-il, s'il sont aussi rapaces que moi, mon avenir est dans ma préretraite. Voila déja une bonne amorce pour de la relaxation. Les oscillations, j'entends. D'un côté on bourre une jeunesse pour assurer soi-disant son avenir et de l'autre cette apparence de connaissance va induire une résistance dûe à la peur du "vieux" pour le "jeune", celui contre lequel il va falloir résister.

Mais ce bourrage de crâne, de quelle nature est-il ? c'est dans cette question que s'implique le pédagogue. Il suffit d'inventer l'infini à partir du fini . imaginons un rectangle métaphorique bien délimité et choisissons un point sur son pourtour, à partir de ce point on

s'aperçoit qu'on peut tracer à l'intérieur autant de chemins qu'on le veut, aussi longs qu'on le veut, qui ne se couperont peut-être pas, mais dont on ne s'apercevra même pas qu'ils se coupent dans le cas contraire. Nos enseignements ressemblent de plus en plus à un écheveau embrouillé, alors qu'on pourrait penser qu'une attitude un tant soit peu scientifique consisterait plutôt à observer le rectangle d'en haut. On accable l'étudiant de myriades de cours dont la plupart ne devraient être que les simples applications à titre d'exercice, de quelques idées fondamentales.

Mais il semblerait que notre préoccupation soit davantage d'occuper le jeune pour lui éviter de penser trop, que de lui apporter la véritable connaissance qui pourrait à la rigueur lui être utile en même temps qu'à la société.

Dans les entreprises "de pointe" le local est de plus en plus organisé comme une prison avec primauté absolue du policier qui contrôle les allées-venues. Ne faut-il pas protéger le sacro-saint "secret" ? Et puis on peut imaginer que cela ne déplaît pas au "chef" qui pense tenir ainsi bien ses troupes en main. Le "vivier" de chômeurs fait le reste.

Le labeur exténuant du dix-neuvième siècle, âge d'or de l'industrie, qui sécrétait tuberculose, alcoolisme et toutes sortes de déchéances, pour le bon motif cela va de soi, est remplacé par un autre labeur, en blouse blanche celui-là, qui s'apparente beaucoup à la vie carcérale. Mais, objecterez vous, en dehors du travail le sujet est libre, et il participe à la vie moderne combien exaltante. Certes s'il n'est pas par hasard possesseur d'un embryon de secret, auquel cas c'est nuit et jour qu'il aura un policier à ses trousses. N'oublions pas les Rosenberg.

Nous devons préparer à la vie de groupe, au travail en équipe, cela ne paraît pas une mince contradiction avec l'obligation de la solitude devant la copie d'examen. L'étudiant peut-il oublier qu'il est en compétition de fait avec ses camarades, même si on lui dit le contraire - chômage oblige - . Ni la société ni l'éducation ne sont faites pour préparer à la vie en société, et plus cette société s'enrichit et la démographie s'accroît plus les hommes se sentent isolés et rejetés. Au lieu d'utiliser les richesses pour ouvrir le passage aux nouveaux venus, elles servent au contraire pour rendre les barrages plus efficaces, on ne lâche vraiement que le minimum pour ne pas donner de mauvaises habitudes. La notion d'examen ou de contrôle ne me paraît pas, en la circonstance jouer un rôle innocent. Ce n'est après tout qu'un aspect de la loi du marché, on ouvre "démocratiquement" les universités au plus grand nombre , mais on augmente dans une proportion au moins équivalente la quantité d'embûches.

Dans le but d'améliorer la qualité du "produit" ?

Comment peut-on de la sorte espérer retenir les plus créatifs, les plus imaginatifs, qui ne se satisfont guère d'un jeu abstrait coupé du réèl,

finalement toujours semblable à lui-même; alors qu'on pourrait raisonnablement penser que ce seront eux les plus utiles pour se débarrasser des vieilles lunes. Pour le côté relaxation, l'expérience est facile à faire. Il suffit de prendre un groupe d'étudiants et de faire croître le nombre de contrôles au cours de l'année. A partir d'un certain point, quelques jours avant chaque examen les étudiants dépassés jettent tout ce qui leur reste de forces dans sa préparation, pour gagner du temps ils sèchent les cours, ce qui a pour effet d'accroître leur travail pour l'examen suivant. Et de la sorte ce phénomène s'amplifie quand la fin de l'année se rapproche.

Solutions ?

Soit on réduit la proportion des contrôles pour laisser le temps de l'absorption et de l'échange de vues, soit on pousse l'étudiant à "l'amélioration de son rendement", en usant un peu de culpabilisation.

Devinez la solution choisie.

Comme la vitesse d'assimilation a du mal à grandir, le "faire semblant" a de l'avenir.

Si là-dessus s'ajoute une diversion du style jeu de la représentativité, présentée comme une chose importante, c'est alors un raz-de-marée qui emporte le tout. Il est tellement plus amusant de jouer au marchand de vent. Il faut d'ailleurs reconnaître que c'est bien la denrée dont la côte ne cesse de croître.Voila une remarque fort peu exploitée par les pédagogues. Ce qui est attrayant est tellement plus facile à assimiler. Mais non, notre civilisation ne peut admettre d'un travail qu'il soit qualifié de sérieux s'il est autre chose qu'un insupportable effort douloureux dans une discipline d'autant plus importante qu'elle est aride et de présentation ennuyeuse.

L'apprentissage des langues vivantes est naturellement un jeu et un véritable plaisir, mais il devient une torture dans l'enseignement officiel, où l'invention de la notion de grammaire entre autres, remplace la joie de l'expression par la contrainte du par-cœur genre table de multiplication. Le raisonnement mathématique sorte de jeu qui consiste à observer la nature d'un point de vue très grossier, à en tirer des propriétés très simplifiées et à établir à partir de là des suites d'évidences, a été confondu avec une sorte de pensée pure imaginaire, et qu'on a tenté de faire passer pour la matérialisation de la "vérité". Ce n'est devenu qu'un moyen formel de plus pour bâtir des critères d'accession à la "caste". Là encore relaxation; front montant lors de l'éviction du latin qui jouait encore ce rôle vers les années cinquante. Et puis chute molle en ce moment car on s'est aperçu que cela ne résolvait rien. Forts d'une expérience perpétuellement inutile recommencera-t-on ce petit jeu avec l'informatique ?

Technocratie aidant on est partis pour remplacer tout bonnement tout cela par les petites lubies du lobby au pouvoir, rentabilité oblige, il faut choisir le bon instant pour employer le mot culture.

Aussi la leçon, la vraie, est-elle parfois bien apprise, qui affûte les dents aux jeunes loups. Gare à celui qui va réussir. Mais sans doute

est-ce bien cela dont on a besoin pour résoudre les crises ? Herriot le libre penseur, disait que la culture c'est ce qui reste quand on a tout oublié. La sémantique de cette idée pourrait sans dommage prendre la forme suivante : on a réussi sa culture quand on a compris ce qu'on vous cachait ,et qui est important et qu'on a oublié ce qu'on vous a appris et qui est sans grand intérêt.

SYSTEME UNIVERSEL

E. Bianco

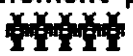
C.R. Subject Classification Informatics: C53 C54 D31 D41

Résumé.

A l'exception du calcul de l'image définitive du sous-système dans les tablettes du système, le système tel qu'il a été défini est complet. J'ai rajouté également un exemple de machine de distribution, une machine de Dupuy. On peut donc déja se faire une idée précise sur l'existence d'un tel type de système.

SYSTEME UNIVERSEL

Je reprends dans son ensemble, le système déja décrit partiellement. A l'exception de la procédure destinée à créér l'image définitive du sous-système dans les tablettes du système, tout est présent. De plus je complète en rajoutant un cas particulier de machine de distribution que j'appellerai machine de Dupuy (cf. Bull. N° 15, p 5). Bien sur il faudrait également programmer tous les calculateurs d'échanges qui sont en relation avec la machine de Dupuy. Mais chacun des algorithmes ainsi obtenus dépend étroitement de la nature du dispositif concerné et ceci dépasse le cadre de cet exposé que je limite volontairement à la démonstration d'existence d'un système universel tel qu'il est défini plus haut.



```
        section init INITIAL ;

    Message := initial ;
    écriture := ultime ;
    [écriture := "dem" , "out" , N° écran , "fconnu" , 2 , - , "file" , - ;

itère INJONCTION ( condition [écriture,etsys = "terminé" ) ;
    répétition INJONCTION ;

    Phrase := ultime ;
    lecture := ultime ;
    [lecture := "dem" , "in" , N° clavier , "fconnu" , 3 , 0 , "ligne" , - ;
    Calcul := initial ;
    [Calcul,Nombre := 0 ;
    [calcul,état := "faux" ;

itère SELECT ( condition [calcul,état = "vrai" ) ;   Compilation du nombre de
                                                     terminaux à sélecter.
    aiguillage ARRET ( [calcul,état = ( "vrai": ARRT , "faux":COMP ) ) ;

COMP : itère LECT ( condition [lecture,etsys = "term" ) ;
    répétition LECT ;


    [Phrase,report := [Phrase,lu ;
```

```
        [lecture,etsys := "dem" ;
        [calcul,Nombre := [calcul,Nombre * 10 ;

aiguillage NB ( [Phrase,report = ( 0:e0 , 1:e1 , 2:e2 , 3:e3 , ... , 9:e9 , ";":e10 )) ;

        e0 : sortie e0 ;

        e1 : [calcul,Nombre := [calcul,Nombre + 1 ;
             sortie e1 ;

        e2 : [calcul,Nombre := [calcul,Nombre + 2 ;
             sortie e2 ;

        e3 : [calcul,Nombre := [calcul,Nombre + 3 ;
             sortie e3 ;
        - - - - - - - - - - - - - - - -
        - - - - - - - - - - - - - - - -
        e9 : [calcul,Nombre := [calcul,Nombre + 9 ;
             sortie e9 ;

        e10 : [calcul,état := "vrai" ;
             sortie e10 ;
             nœud NB ;

        sortie COMP ;

ARRT : sortie ARRT ;
        nœud ARRET ;

        répétition SELECT ;

        Phrase := initial
        PILOTE := initial ;
        lecture := initial
itère LANCE ( [calcul,Nomb1 de 1 à [calcul,Nombre ) ;
        [PILOTE,Nterm := [calcul,Nomb1 ;
        [PILOTE,Etach := "horstxt" ;
        [PILOTE,Ech := "in" ;
        PILOTE := → PILOTE ;
        [lecture := "dem" , "in" , N° clavier , "fconnu" , 3 , - , "file" , - ;
        lecture := → lecture ;
```

Le nombre de terminaux étant connu on prépare la file PILOTE en attente "horstxt" pour que le système puisse déjà commencer à l'explorer.

```
        [Phrase,er := 0 ;
        Phrase := → Phrase ;

        répétition LANCE ;

        Dispon := initial ;
        [Dispon,caract1 := "fin" ;
        GRILLEFIN := initial ;
        LISTFIN := initial ;

        ligne LISTFIN , 1 ;          (Avant de charger un quelconque sous-
                                          système)
fin section init SURVISEUR ;

        section SURVISEUR ;

itère PERPET ;
        PILOTE := [calcul,Nombre → initial ;
        PILOTE := ← PILOTE ;

itère TACHE ( PIL de initial à PILOTE ) ;

        lecture := [PIL,NTERM → initial ;
        lecture := ← lecture ;

aiguillage COMMUNICATION ( [PIL,ech = ( "in":Lect , "out":Ecrit ) ) ;

        Lect : aiguillage LECTURE ( [lecture,etsys = ( "dem":suite , "en cours":suite,
                                , "lu":traitement ) ) ;

                suite : sortie suite ;

                traitement : Phrase := [PIL,Nterm → initial ;
                        Phrase := ← Phrase ;
                        [Phrase,report := [Phrase,lu ;
                        [lecture,etsys := "dem" ;
```

```
aiguillage SEPAR ( [PIL,ETACH = ( "horstxt":HORSTXT , "ssys":SOUSS ) ) ;

        HORSTXT : insérer HORSTEXTE ;
                insérer proteste ;          Emission d'un message d'er-
                sortie HORSTXT ;            reur si [Phrase,er ≠ 0 .

        SOUSS : TSS := [PIL,AX → initial ;  Travail sur le VIIF.
                DISF := [TSS,3 → TSS ;
                DISC := [TSS,4 → TSS ;

aiguillage FONCTION ( [TSS,2 = ( "en cours":calcul , "terminé":stop ) ) ;

        calcul : insérer QUANTUM ;
                sortie calcul ;


        stop : insérer SUPPRESS ;       Libère la place occupée par le sous-
                sortie stop ;           système, et repasse en mode hors-texte.

                nœud FONCTION ;

                        sortie SOUSS ;
                        nœud SEPAR ;

                sortie traitement ;
                nœud LECTURE ;

        sortie Lect ;

        Ecrit : écriture := [PIL,NTERM → initial ;
                écriture := ← écriture ;
        aiguillage Sortmess ( [écriture,etsys = ( "dem":suitecr , "en cours":suitecr,
                                                  "term":Finecr ) ) ;

                suitecr : sortie suitecr ;

                Finecr : [écriture := "dem" , "in" , [PIL,NTERM , "fconnu" , 3 , 2 ,
"ligne" , - ;
                        [PIL,ech := "in" ;      A titre d'exemple je dirai que:
                        sortie Finecr ;         le numéro du clavier est égal à
                                                        [PIL,NTERM
                        nœud Sortmess ;         et celui de l'écran qui lui correspond
```

$[_{PIL,NTERM} + 100$

sortie Ecrit ;

nœud COMMUNICATION ;

insérer FAUTE ; Emet les messages d'erreur des sous-systèmes

répétition TACHE ;

répétition PERPET ;

section FAUTE ;

aiguillage ERREURCOMP ($[_{DISC,9}$ = ("dem":ERR sinon CONT)) ;

CONT : sortie CONT ;

ERR : POOLC := $[_{DISC,5} \rightarrow$ TSS ;
POOLF := $[_{DISC,7} \rightarrow$ TSS ;

itère RECHFILE (POOL de POOLC à POOLF condition $[_{DISC,16}$ = $[_{POOL,7}$) ;
répétition RECHFILE ;

S1 := $[_{POOL,5} \rightarrow$ initial ; La file des messages est repérée par son rang sur la file support.
mess := interne S1 , 1 ; Calcul de la référence interne de la file support: mess qui permet d'atteindre l'intérieur de la file supportée.
ligne mess , 1 ;
messf := ultime S1 ;

itère MESSERR (messc de mess à messf condition $[_{DISC,17}$ = $[_{messc,1}$) ;
ligne messc , 1 ;
répétition MESSERR ;

itère CARTOUH (cartc de initial à ultime condition $[_{cartc,1}$ = "nil") ;
répétition CARTOUH ;

```
            écriture := [PIL,NTERM → initial ;
            écriture := ← écriture ;
            [calcul,nomb1 := [PIL,NTERM + 100 ;
            [écriture := "dem" , "out" , [calcul,nomb1 , "support" , rang SI , rang
messc , "ligne" , text ;
            [PIL,ech := "out" ;
            [DISC,9 := "nil" ;

      sortie ERR ;

      nœud ERREURCOMP ;

            fin section FAUTE ;
```

```
section QUANTUM ;

      aiguillage INOUT ( [DISC,10 = ( "dem":prep , "en cours":suite ,
"terminé":Achève , "nil":reprise ) ) ;

      prep : carts := ultime ;

itère CHCARTOU ( cartc de initial à carts condition [cartc,état = "nil" ) ;
      répétition CHCARTOU ;

      [DISC,15 := rang cartc ;        Il faudrait vérifier le cas où rien
      [DISC,10 := "en cours" ;        n'est libre.

      sortie prep ;

      suite : sortie suite ;

      Achève : [DISC,10 := "nil" ;
            insérer COMPUT ;
            sortie Achève ;

      reprise : insérer COMPUT ;
```

```
        sortie reprise ;

        nœud INOUT ;

fin section QUANTUM ;
```

***** *****

```
section COMPUT ;

    aiguillage DEROULT ( [DISC,4 = ( "dem":START , "en cours":START ,
"terminé": retsys , "suite":compsuiv ) ) ;

    START : DIS := [TSS,4 → TSS ;
        insertion fractionnée DIS ;
        sortie START ;

    compsuiv : DISK := [DISC,7 → TSS ;   L'élément en FR.
        insérer CALCPARAM ;
        [TSS,4 := [DISC,6 ;              Numéro du nouveau compilateur.
        DISC := [TSS,4 → TSS ;           Préparation du VIIF du nouveau
        [DISC,4 := "demandé" ;           compilateur.
        sortie compsuiv ;

    retsys : [TSS,2 := "terminé" ;
        sortie retsys ;

        nœud DEROULT ;

fin section COMPUT ;
```

***** *****

```
section SUPPRESS ;

        POOL := [TSS,3 → TSS ;
        POOL := → POOL ;
```

```
        DISK := [DISC,7 → TSS ;
        DISK := ← DISK ;
    itère VIDAGE ( POOLC de POOL à DISK ) ;

    aiguillage CAS ( [POOLC,1 = ( "fin":RAF sinon VIRER ) ;

    RAF : sortie RAF ;                          Opération qui consiste à
                                                placer les rangs des files
    VIRER : libres := [POOLC,5 → initial ;      désormais inutiles, dans
        conf := rang libre → initial ;          la file "dispon" afin d'être
        libérer conf ;                          réutilisées à l'occasion
        Dispon := → Dispon ;                    d'une création de tâche,
        [Dispon,lieu1 := rang libres ;          ou d'une création de file,
        sortie VIRER ;                          sur une instruction d'arrêt.

        nœud CAS ;

    répétition VIDAGE ;

        DISK := [DISC,7 → TSS ;
        DISK := [DISK,4 → DISK ;

    itère ELIMIN ( POOLC de TSS à DISK ) ;
        condenser POOLC ;
        répétition ELIMIN ;

    fin section SUPPRESS ;
```

```
section CALCPARAM ;

        POOLC := → DISK ;
    itère Occupation ( [calcul,Nomb1 de 1 à [DISK,4 ) ;
        POOL := [DISC,5 → TSS ;

        itère CHERCH ( condition [POOLC,7 = [POOL,7 ) ;
        POOL := → POOL ;                Cherche la file externe en pool des
        répétition CHERCH ;             files, sur le numéro de la file.
```

```
aiguillage MODEFILE ( [POOLC,5 = ( "à créer":CREER , "réduction":MODIF ,
"extension":MODIF ) ) ;

CREER : aiguillage DISP ( [Dispon,caract1 = ( "fin":Occuper ,
"libre":REOccuper ) ) ;
        REOccuper : libres := [Dispon,lieu → initial ;
                Dispon := ← Dispon ;
                Supplib := rang libres → initial ;
                file Supplib,POOLC ;
                [POOLC,5 := rang supplib ;
                [POOLC,2 := [POOL,2 ;
                [POOL := [POOLC ;

        sortie REOccuper ;

        Occuper : conf := rang libre → initial ;
                file conf, [POOLC,4 ;
                [POOL := [POOLC ;
                [POOL,5 := rang libre ;
                [libre,caract := "ω" ;
                occupé := libre ;
                libre := → libre ;

                sortie Occuper ;

        nœud DISP ;

                sortie CREER ;

        MODIF : libres := [POOL,5 → initial ;
                Supplib := rang libre → initial ;
                conf := rang libre → initial ;
                file conf, [POOLC,4 ;
                [conf := [supplib ;
                libérer Supplib ;
                file Supplib, [POOLC,4 ;
                [supplib := [conf ;
                libérer conf ;
                [POOLC,5 := [POOL,5 ;
```

```
[POOL := [POOLC ;

sortie MODIF ;

nœud MODEFILE ;

répétition Occupation ;

fin section CALCPARAM;
```

COMMENTAIRES

Je présente ici une version de toute évidence simplifiée d'un type de système dit "d'utilisation collective en temps partagé" . Toute autre forme de système pourrait être construite de la même manière.

Dans la section INITIAL on réalise une fois pour toutes la mise en œuvre du travail, lors de la mise sous tension de l'ordinateur.

Un message est émis pour demander le nombre de terminaux à prendre en compte. Ces terminaux seront numérotés de 1 à n , n est la valeur fournie par le responsable du système sur une console numérotée 0. C'est la boucle INJONCTION qui émet ce message.

L'itère SELECT compile la valeur n .

L'itère LANCE construit la file en PIL,PILOTE qui contient les images permanentes des terminaux. Chacune de ces images renvoit à une image temporaire qui est l'image de la tâche en cours.

La section INITIAL renvoit à la section SURVISEUR qui contient la boucle définitive sous la forme de l'itère PERPET.

L'itère TACHE explore la suite des images de terminaux. C'est l'itère PERPET qui rend cette boucle répétitive.

Pour chaque image de terminal s'effectuent, dans l'ordre, les choix suivants.

L'état normal pour un terminal c'est la lecture d'une lettre. Mais exceptionnellement on peut être en émission de message pour le compte du système dans le cas d'erreur ou de situation particulière. Ceci représente la partie conversationnelle du système, qui peut être également amené à émettre un message issu d'un sous-système.

L'aiguillage COMMUNICATION distingue l'état exceptionnel d'émission de message de l'état normal de lecture d'une lettre. Dans ce cas,on vérifie qu'une lettre a été bien lue, sinon on passe à l'mage suivante.

Si la lettre a été lue l'aiguillage SEPAR distingue le traitement du HORSTEXTE du SOUS-SYSTEME.

Dans le premier cas on insère le compilateur HORSTEXTE.

Dans l'autre cas, donc pour le sous-système, l'aiguillage FONCTION détermine si la tâche est achevée,

auquel cas la section SUPPRESS libère la place occupée par son VIIF .

Et si la tâche est toujours en cours, on insère QUANTUM.

A partir de cet instant on traite une tâche du sous-système. Soit il y a échange de masse, soit il y a calcul. C'est l'aiguillage INOUT qui distingue.

Dans le cas où il n'y a pas échange, $[_{DISC,10}$ = "nil" , alors on insère COMPUT. Si l'état de la tâche est, "demandé" ou "en cours", alors on réalise l'insertion fractionnée. Si l'état de la tâche est "terminé", on insère CALCPARAM dont le travail consiste à modifier la structure des files à la demande des compilateurs. Il s'agit là d'un travail de gestion de la place en mémoire, extension de files, restriction à la seule place occupée, etc.

TRAITEMENTS DE TRANSITION

Lors d'un retour de l'affectation d'un quantum de travail, la section FAUTE détecte la faute éventuelle signalée par le compilateur et prépare l'émission d'un message issu d'une **file valeur** . On signale au système, par

$[_{PIL,ech}$ = "out"

qu'un message est à émettre, le cartouche étant préparé. C'est l'aiguillage COMMUNICATION de la section SURVISEUR qui, lors du balayage de la tâche, surveille l'émission en "Ecrit : ".

La section SUPPRESS a pour rôle de récupérer la place occupée par l'image de tâche, essentiellement le VIIF. Pour cela j'utilise une file supplémentaire à rajouter au contexte déclaré Bulletin N° 22.

5000 tableau (caract1(1) , lieu1(2) , descr1(2)) : Dispon : Disponc ;

Ce tableau sert à noter les rangs des files libérées afin de pouvoir les réutiliser lors de la création de nouvelles tâches.

La sémantique de l'instruction libérer prévoit la réutilisation de la place occupée par les valeurs, alors que l'accès réservé dans la directory reste disponible.

FORMALISATION DES ECHANGES

Dans une dernière mise au point on va préciser la nature de l'information à transmettre pour un fonctionnement cohérent de la machine de distribution. Il faut d'abord remarquer que dans un cartouche, unité naturelle de communication entre système et machine de distribution, on va devoir porter des informations qui concernent des files décalées sémantiquement entre elles.

En effet on aura à échanger des contenus de files déclarées dans le système, donc connues à l'avance, et des contenus de files construites ultérieurement et connues alors

uniquement comme des files dans une file support.

Dans le cartouche de l'unité centrale je porte donc la suite des informations suivantes:

état , sens , numéro du dispositif d'échange , sorte de file , rang file , rang ligne , choix file ou ligne , nom logique de l'enregistrement ;

numéro du dispositif d'échange : il s'agit d'un numéro logique attribué arbitrairement une fois pour toutes.
sorte de file = "fconnu" / "support" .
choix file ou ligne = "file" / "ligne" .

La construction de la machine de distribution exige la conception de deux nouveaux types de files. Il faut en effet pouvoir disposer de mémoires spécifiques de la machine de distribution mais aussi des mémoires de chacun des calculateurs d'échanges. Je renvoie à l'article **Machine de distribution** du bulletin N° 17. Dans cet article je décrivais une machine en utilisant le modèle procédure formelle, qui donnait une idée de la structure électronique. Je vais maintenant construire une machine de distribution à l'aide d'un modèle procédure formelle symbolique, donc plus abstrait, mais dont les propriétés sémantiques demeurent fondamentalement les mêmes.

A partir du moment où je connais le liste des dispositifs d'échange que je note dans une file en "DEdisp" de la machine de distribution, il me faut disposer de deux autres files qui appartiennent aux calculateurs d'échanges. L'une contient une image de cartouche, une seule par calculateur suffit. L'autre est destinée à contenir les valeurs en transit. Je définis ainsi un cartouche de type "échange" et une dynamique également "échange" pour indiquer que ces files appartiennent à tous les calculateurs d'échanges à raison d'une ligne par dispositif. Pour des raisons de commodité, le numéro logique du dispositif est pris comme rang de la ligne qui lui correspond.

Si un calculateur d'échange contrôle plusieurs dispositifs il dispose d'autant de lignes dans chacune de ces deux files.

400 cartouche échange (et1(1) , sens1(1) , tDE(2) , vol(2) , retE(2)) : Ech : Echc;

20 000 , 400 dynamique échange (lc) , 1(char(1)) : mot : lettre ;

400 valeur tableau distribution (typ1(1) , NumCE(2) , occ(1) ,) : DEdisp : DEdispc , ("DK", 301,-,-) , ("DK",302,-,-) , ("DK",303,-,-) , ... , ("RM",321,-,-) , ("RM",322,-,-) , ... ;

A ces files je rajoute les trois files suivantes à titre de complément:

6 : 62,3 valeur dynamique (car) , 1(merr(1)) : Rlet , ("soussyst" en trop) , ("intégré" en trop) , (Ce sous-système n'existe pas) ;

4 : 100 cartouche (etsys(1) , sensys(1) , NDE(2) , tyfsys(1) , Nefsys(2) , rangsys(2) , modsys(1) , logsys(1)) : lect : lecture : écriture ;

5 : 50 cartouche (état(1) , sens(1) , NdeDE(2) , Tfil(1) , Nfil(2) , rangl(2) , modf(1) , log(2) : cart : carte : carts : cartc ;

Les deux dernières ne sont que des modifications apportées aux noms en tenant compte de la construction de la machine de distribution.

Les trois files consacrées à l'échange se correspondent ligne à ligne de telle manière qu'un même rang permette d'accéder à tout ce qui concerne un même dispositif.

Il faut compléter cet ensemble par une propriété qui, bien que très générale, se présente ici sous une forme un peu spécifique. J'ai pris en compte le côté conversationnel, convivial, en séparant les échanges en deux parties, ceux qui sont spécifiques du système et ceux qui vont dépendre exclusivement de la nature des sous-systèmes, et qu'on ne peut donc prévoir en détail à l'avance. Mais ce qui est un peu à cheval sur les deux c'est en fait le captage d'information conversationnel qui, dans ce genre de système est placé sous contrôle du système, ce qui assure le meilleur service puisqu'on corrige le texte au fur et à mesure, mais introduit un degré de complexité supplémentaire dans la mesure où la réponse qui incombe forcément au système ne peut lui être que fournie par un sous-système. Une normalisation devient indispensable que l'on inclut dans le langage.

LA MACHINE DE DISTRIBUTION

```
sectionMD DISTRIBUTION ;
                DEdisp := initial ;

    itère PERPETECH ;

        itère ECHTOT ( carte de initial à ultime ) ;

        itère CONVERSA ( lecture de initial à ultime ) ;

    aiguillage DEMD ( [lecture,etsys = ( "dem" : Lance sinon Autre ) ) ;

        Lance : insérer ECHCONV ;
```

```
                sortie Lance ;

            Autre : sortie Autre ;
                nœud DEMD ;

            répétition CONVERSA ;

        aiguillage DEMASSE ( [carte,état = ( "dem":MOUV , sinon STAT ) ;

            MOUV : insérer ECHMASS ;
                sortie MOUV ;

            STAT : sortie STAT ;
                nœud DEMASSE ;

        insérer FINAL ;

            répétition ECHTOT ;

            répétition PERPETECH ;

fin section MD DISTRIBUTION ;
```

Je choisis arbitrairement d'explorer entièrement le cartouche conversationnel en "lecture-écriture", pour chacun des éléments du cartouche d'échanges de masse en "carte", afin d'assurer le maximum de priorité au conversationnel.

Tout autre mode de trafic pourrait être choisi sans rien changer au fait fondamental.

```
section ECHCONV ;

            Ech := [lecture,NDE → initial ;
            [Ech,et1 := "dem" ;
            [Ech,sens1 := "in" ;
            [lecture,etsys := "en cours" ;

fin section ECHCONV ;

section ECHMASS ;

            aiguillage Sortefile ( [carte,Tfil = ( "support": SUPP , "fconnu":
CONNU ) ) ;

        SUPP : Ech := [carte,NdeDE → initial ;
            [Ech,et1 := "dem" ;
```

```
        [Ech,sens1 := [carte,sens ;
        [Ech,tDE := [carte,NdeDE ;
        [Ech,retE := rang carte ;
        P := [carte,NFIL → initial ;
        mot := [Ech,tDE → initial ;

aiguillage LIGNE ( [carte,modf = ( "ligne":MOVLIG , "file":MOVFIL ) ) ;

MOVLIG : mess := nom P ;                          Déplacement d'une ligne.
        mess := [carte,rangl → mess ;
        [Ech,vol := volume mess ;

        aiguillage MOUVTL ( [carte,sens = ( "in":Oncont , "out":Extrac ) ) ;

                Oncont : sortie Oncont ;

                Extrac : S := 1 ;
                itère SORTIEL ( ic de 1 à [Ech,vol ) ;
                        [mot,ic := [mess,S ;
                        S := S + 1 ;
                répétition SORTIEL ;
                        sortie Extrac ;

                        nœud MOUVTL ;

        sortie MOVLIG ;

MOVFIL : [Ech,vol := volume P ;                   Déplacement d'une file.
                        S := 1 ;
        itère SORTIEF (ic de 1 à [Ech,vol ) ;
                [mot,ic := [P,S ;
                S := S + 1 ;
        répétition SORTIEF ;

                        sortie MOVFIL ;

                        nœud LIGNE ;

                sortie SUPP ;

CONNU : Ech := [carte,NdeDE → initial ;
```

```
aiguillage NUMFILE ( [carte,Nfil = ( 2:DEUX , 3:TROIS , 5:CINQ , 6:SIX );

        DEUX : [Ech,et1 := "dem" ;
               [Ech,sens := "out" ;
               [Ech,tDE := [carte,NdeDE ;
               [Ech,vol := 42 ;
               sortie DEUX ;

        TROIS : [Ech,vol := 1 ;
               [Ech,et1 := "dem" ;
               [Ech,sens := "in" ;
               [Ech,tDE := [lecture,NDE ;
               [Ech,retE := rang lecture ;
               sortie TROIS ;

        CINQ : [Ech,vol := 1 ;
               [Ech,et1 := "dem";
               [Ech,sens := "in" ;
               [Ech,tDE := [carte,NdeDE ;
               [Ech,retE := rang carte ;
               sortie CINQ ;

        SIX : Rlet := [carte,rang] → initial ;
               [Ech,vol := volume Rlet ;
               [Ech,et1 := "dem" ;
               [Ech,sens := "out" ;
               [Ech,tDE := [carte,NdeDE ;
               sortie SIX ;

        nœud NUMFILE ;

        sortie CONNU ;

        nœud SORTEFILE ;

fin section ECHMASS ;

section FINAL ;
```

```
aiguillage ACHEVE ( [DEdisp,occ = ( "term":TERM sinon TOUR ) ) ;

        TERM : aiguillage MASCONV ( [Dispo,typ = ( "convers":CONV ,
"masse":MASS ) ) ;

                CONV : Echc := rang DEdisp → initial ;
                       écriture := [Ech,retE → initial ;
                       [écriture,etsys := "term" ;

        aiguillage ENTSORT ([écriture,sensys = ( "in":ENTIN , "out":ENTOUT ) ) ;

          ENTIN : mot := rang DEdisp → initial ;
                  Phrase := rang écriture → initial ;
                  [Phrase,lu := [mot ;
                  sortie ENTIN ;                FINAL recherche les traitements
                                                achevés, dans les cartouches des
          ENTOUT : sortie ENTOUT ;              dispositifs d'échanges. Il ne traite
                  nœud ENTSORT ;                qu'un seul cas à la fois.


          sortie CONV ;

                MASS : mot := rang DEdisp → initial ;
                       Echc := rang DEdisp → initial ;
                       cart := [Echc,retE → initial ;

          aiguillage ENTSORTM ( [carte,sens = ("in":ENMIN , "out":EMOUT ) ) ;

                ENMIN : P := [cart,Nfil → initial ;

aiguillage FILIGN ( [cart,modf = ( "file":EXTRFIL , "ligne":EXTRLIG ) ) ;
                       EXTRFIL : S := 1 ;

                       itère VALFIL ( lc de 1 à [Echc,vol ) ;
                              [P,S := [mot,lc ;
                              S := S + 1 ;
                       répétition VALFIL ;

                              sortie EXTRFIL ;
```

```
                    EXTRLIG : messc := nom P ;
                          messc := [cart,rang] → messc ;
                          S := 1 ;
                itère VALIGN (ic de 1 à [Echc,vol ) ;
                    [messc,S := [mot,ic ;
                    S := S + 1 ;
                    répétition VALIGN ;

                          sortie EXTRLIG ;

                    nœud FILIGN ;

                [cart,état := "terminé" ;

                    sortie ENMIN ;

                EMOUT : [cart,état := "terminé" ;
                    sortie EMOUT ;

                nœud ENTSORM ;

          sortie MASS ;

          nœud MASCONV ;

     sortie TERM ;

          TOUR : sortie TOUR ;

               nœud ACHEVE ;

          DEdisp := → DEdisp ;
     aiguillage INCREM ( [DEdisp,typ = ( "*":Init sinon Tourne ) ) ;
               Init : DEdisp := initial ;
                    sortie Init ;
               Tourne : sortie Tourne ;
                    nœud INCREM ;

fin section FINAL ;
```

COMPILATEUR DE MACHINE UNVERSELLE: MU.

Christophe Bisière,
Denis Iwanesko,
Jean Michel Knippel,
Jean Luc Massat.

C.R. Subject classification informatics. : D.3

"Imagination - C'est cette partie dominante dans l'homme (...).L'imagination dispose de tout; elle fait la beauté, la justice, et le bonheur, qui est le tout du monde."

Pascal, Pensées, 82.

Résumé.

Dans le cadre de la formation initiale des Universités d'Aix-Marseille nous avons mis au point un compilateur de la Procédure Formelle, dit compilateur de machine universelle,
capable de produire un code directement exécutable par un ordinateur "Mu"
tel est son nom. Toutefois, afin d'implanter le compilateur sur différents modèles de machine d'une même classe, nous avons adjoint un module " dérouleur-exécuteur" du langage sortie du compilateur.
La simplicité de forme langagière de la machine de Nolin est utilisée. Les mécanismes de base de l'informatique et leur présentation sont d'autant plus clairs: capacité à faire de l'arithmétique, aboutir à la construction d'algorithmes fournisseurs de procédures variées: manipulateurs de fichiers, éditeurs de texte, langage de programmation, compilateur, système, etc...
Après une présentation du compilateur, des modifications sur le langage de la procédure formelle que nous avons effectuées; le lecteur curieux trouvera quelques exemples de programmation concernant la récursivité et un éxécuteur de code qui ont été testés avec succès par "MU".

COMPILATEUR DE MACHINE UNIVERSELLE: MU.

Ch. BISIERE, D. IWANESKO, J.M.KNIPPEL, J.L.MASSAT.

I. Préliminaires:

L'article qui suit est la description d'un compilateur de langage de programmation destiné à aborder à la fois l'architecture des processeurs et leur utilisation à l'aide de la Machine de Nolin. Les critères pédagogiques d'enseignement et de formation à la recherche ont été formalisés par Monsieur le Professeur E.Bianco (1), (7). L'étude et la réalisation du processeur expérimental MCA-0 et de ses successeurs a été faite au sein de l'Université (1), (2), (5), est venue ensuite la phase de développement des programmes aidant l'utilisateur à plus d'un titre. Ils sont pour lui fournisseurs de procédures utilitaires, manipulateurs de fichiers, certains spécialisés. La notion de système logiciel et de système machine a été étudiée (4), il restait à diffuser un compilateur (8). L'utilisateur peut alors à loisir se lancer dans la concrétisation de quelques fondements informatiques au détour de quelques lectures conseillées (3), (6), que sont la réduction des machines à une forme la plus compacte possible ou la notion d'insertion de code. Ceci n'est qu'une idée de cheminement; l'exemple traité de la récursivité donnera une autre approche de mécanisme.

Il s'agit d'un travail collectif qui déborde largement des cloisonnements, formation initiale ou continue et D.E.A., ou formations doctorales, puisque suivant le niveau où l'on se situe on pourra développer ou seulement utiliser une notion, un concept. Il faut noter que nous n'aurions pas fait cette réalisation sans les conseils éclairés de Monsieur le Professeur E.Bianco. Nous pourrions qualifier le produit actuel d'expérimental, toutefois la mise à disposition aux étudiants de DEUG 1° Cycle de l'Université d'Aix-Marseille II, de MMIAGe 2° Cycle Programme des Universités d'Aix-Marseille II-III a donné des résultats positifs et nous n'avons pas encore relevé d'erreurs dans la version actuelle du produit.

II. Présentation générale du compilateur:

Le compilateur "MU" et le dérouleur "DE" ont été développés entre 1986 et 1987, leurs réalisations initiales ont demandé 1 homme-année (8).

Le degré d'abstraction du langage à compiler ici rend la traduction en langage "machine" plus simple. Nous avons choisi comme langage de programmation "LP", le langage de programmation de la procédure formelle. Cela signifie que nous avons introduit une structure complète de programmation (10).

Il nous faut alors pouvoir avoir le moyen de créer et de récupérer des configurations de données(10). Ce sera le rôle de deux instructions du langage de manipulations de données "LE", que nous avons introduites dans le langage de programmation. Nous développons les deux instructions de ce langage dans la partie de ce paragraphe concernant la facilité de mise au point des programmes.

Il nous reste alors à définir un langage de manipulation en bloc de configurations et de programmes "LC"(10).
La création de programmes et éventuellement des différentes configurations de données sera le travail de l'éditeur hôte du système sur lequel est implanté le compilateur "MU". Si nous prenons l'exemple du système VM/SP-IBM, le produit utilisé sera l'éditeur de textes XEDIT. Enfin, la manipulation des programmes et des configurations comme des objets sera faite par le compilateur "MU", et le dérouleur-exécuteur " DE ". Les objets sont tout simplement des fichiers du système hôte de "MU". Sur un exemple trivial de programme minimum à écrire, nous donnons dans les figures suivantes la schématisation de l'enchainement des différents langages: "LP", "LE", "LC".

Nous laissons le soin au lecteur de trouver ce que fait le programme. Nos premiers lecteurs gagneront des exemplaires de ce numéro!

```
COMPIL3                                          (LC)
Compilateur de procédure formelle                 .
++++++++++++++++++++++++++++++++                  .
nom de fichier en entrée: ESSAI 1 LITAM A        (LC)
*texte source:                          (LP)
        proc vide                                 .
        para 0                                    .
        index 0                                   .
        var 0                                     .
        début
        fin.                                     (LP)
nom de fichier en sortie: ESSAI 1 NOLIN A        (LC)
6 lignes compilées,                               .
0 erreur trouvée.                                 .
longueur du code: 10.                             .
ready; T=0.02/0.06 14.01:56                       .
```

Figure.1. Schéma général de compilation d'un exemple simple.

Le texte en majuscules sur les figures.1. et .2. est celui tapé par l'utilisateur, le texte du fichier est à cette étape supposé déjà présent dans le fichier ESSAI 1 construit par l'éditeur hôte. Le compilateur engendre un langage interprétatif, ici stocké dans le fichier de sortie ESSAI 1 NOLIN A. Ce langage est destiné à être exécuté par un interpréteur, donc par un simulateur de machine qui est ici le dérouleur-exécuteur "DE".

```
DEROUL3                                          (LC)
fichier à exécuter: ESSAI 1 NOLIN A              (LC)
- 00:    10                                      (LP)
- 01:     1                                       .
- 02:    00                                       .
- 03: 8704  0                                     .
- 05: 9216  0                                     .
- 07: 9728  0                                     .
- 09:  512                                       (LP)
ok                                               (LC)
ready; T=0.02/0.08 14:05:41
```

Figure.2. Schéma général du dérouleur et exemple de PROC VIDE.

On demande beaucoup de services au compilateur (9), aussi nous avons accordé une grande attention, entre autre aux points suivants, lorsque nous avons réalisé ces modules:

II.1 fiabilité dans son utilisation:

Les études théoriques et pratiques déjà évoquées çi-dessus (1), (2), (3), (4), (5), (6), (7) nous garantissent l'utilisation d'un acquis déjà considérable sur les plans tant matériel que logiciel.

II.2. rapidité de compilation:

Nous ne discuterons pas ici de la poussière de seconde de plus ou de moins sur telle ou telle machine. L'objectif est d'écrire le compilateur à moindre frais à l'aide d'un langage développé actuellement sur toute machine "petite" ou "grande", choix subjectif du langage de programmation Pascal. Nous avons freiné la prolifération d'utilisation des mots clefs du langage afin que la portabilité du compilateur ne devienne qu'une étape dans le projet et non pas une fin en soi. Peut-être relançons nous ici le "mini-Pascal".

II.3. facilité de mise au point:

Par rapport aux instructions classiques que nous rappellerons dans le paragraphe suivant, nous avons rajouté deux instructions: LIRE et ECRIRE qui permettent de visualiser le contenu d'une variable ou d'en faire la saisie à partir d'un périphérique. Ceci peut paraître simpliste, l'utilisateur pourra ainsi réfléchir aux insertions judicieuses d'instructions de trace.... pour mettre au point ses procédures personnelles.

II.4. efficacité du code engendré:

Nous rappelons que la structure et le fonctionnement des calculateurs est abordée par le concept des machines à jeu d'instructions réduit, ce qui permet d'étudier les réductions

structurelles et logicielles de la machine à cases adressables par exemple.

II.5. clarté de ses messages d'erreur:

En fin de compilation, en cas d'erreur, le code objet ne sera pas sauvegardé. Si l'erreur est fatale la compilation s'arrête et les fichiers standards sont fermés.
Nous donnons ici la liste des erreurs qui éclaireront l'utilisateur:
-caractère hors alphabet,
-erreur de syntaxe,
-constante attendue,
-constante supérieure à 65535,
-étiquette déjà connue,
-instruction inconnue,
-opérateur inconnu,
-instruction mal placée,
-étiquette incorrecte,
-comparateur inconnu,
-procédure inconnue,
-case hors configuration,
-étiquette mal placée,
-étiquette attendue,
-nombre de "vers" en avant trop grand.

Nous rajoutons les erreurs fatales suivantes:
-mémoire pleine,
-fin de programme non trouvée,
-nombre maximum d'étiquettes atteint.

II.6. possibilités de modifier aisément le compilateur:

Nous donnerons l'exemple des "entrées-sorties" et le traitement des fichiers système qui ont été tous transformés en procédures que l'on modifiera à volonté suivant la machine cible.

III. Modifications sur le langage de la procédure formelle:

Le tableau de la figure.3 donne les types d'opérations, le nombre d'opérandes et le code opératoire correspondant. On notera par rapport à la forme retenue par la procédure formelle classique (7), l'adjonction des instructions: LIRE, ECRIRE que nous avons déjà commentées dans le paragraphe précédent. Au sujet des instructions de déclarations de types l'ordre PARA suivi de INDEX et enfin de VAR est obligatoire. Si l'utilisateur ne déclare aucun type, il le fera de la manière donnée dans l'exemple de la figure.1. La syntaxe originelle de VAR, par exemple dans VAR 2:1,1 a été remplacée par VAR 2;1,1 pour éviter la confusion avec le symbole ":" qui signale les labels. Il est à remarquer également que les instructions de manipulation de files ne sont pas prises en compte ici, et qu'une instruction est présente par ligne sans format préétabli. Le seul séparateur d'instruction explicite, apparaît après l'instruction FIN, c'est le symbole "." comme le montre la figure .1.

code opératoire	type d'opérations	nombre d'opérandes
0	début	0
1	fin	0
2	:=+	3
3	:=-	3
4	:=*	3
5	:=/	3
6	:=	2
7	si =	3
8	si/=	3
9	si =	3
10	si	3
11	si=	3
12	si	3
13	vers	1
14	insére	X
15	lire	1
16	écrire	1
17	para	X

18	index	X
19	var	X

Figure.3. Tableau des instructions du langage compilé.

X dans le tableau de la figure précédente signifie que le nombre maximum d'opérandes dépend de la machine support du compilateur. L'instruction d'insertion (6) existe dans le langage, nommée INSERE, nous donnons un exemple d'emploi de cette instruction dans le paragraphe des exemples choisis. Le mécanisme de l'insertion n'est pas rappelé ici, le lecteur se reportera à la littérature donnée en bibliographie. Notons une dernière modification syntaxique des opérandes dans le tableau de la figure.4, où les symboles "(" et ")" ont été utilisés.

forme d'opérande	code
C("constante")	0
C("constante","constante")	1
C(I)	2
C(W)	3
" constante"	4

Figure.4. Syntaxe des opérandes

La valeur maximum d'une "constante" est actuellement de 14999, cela pouvant d'ailleurs être modifié aisément lors de l'implantation du compilateur sur telle ou telle machine.

La constitution du code instruction est la suivante, nous la donnons pour mémoire. L'utilisateur ne devra d'ailleurs pas être désorienté par les divers transcodages dûs aux diverses implantations qui rendent la lecture du code opératoire délicate. Le jeu consiste à le retrouver sur cette base..

poids faible poids fort

type 1 X X

I . . . . . I . . I . I . . I . . I

code opératoire type2 type3

Figure.5. Constitution du code opératoire.

Nous noterons au passage la possibilité d'insérer des commentaires dans le code source du programme. La ligne de commentaire commence par le symbole "*".

IV. Types:

Les objets simples qui peuvent être manipulés par la procédure formelle sont, nous le rappelons, les variables et les constantes. Dans un programme, le type de ces objets doit être obligatoirement déclaré. Un type est une association d'un ensemble de valeurs numériques. Les seuls types numériques connus sont les entiers signés qui varient de -32768 à 32767. Toutefois il faudra conserver à l'esprit la table de transformation des entiers non signés en entiers signés.

-32768	32768
-32767	32769
.	.
.	.
-2	65534
-1	65535
0	0
1	1
.	.
.	.
32767	32767

Figure.6. Tableau de transformation des entiers.

La construction des réels est à faire à partir de ceci, ainsi que les procédures de manipulation qui deviennent des programmes

standard de bibliothèques.

V. Structures de données:

Au début du code deux cases sont réservées par le compilateur. La première indique la longueur du code, c'est à dire l'adresse absolue de la première configuration des données. La seconde donne l'adresse relative moins un du sous programme qui tient lieu de programme principal. Ensuite viennent les trois sortes de variables:

-les paramètres de la procédure (cf. PARA),
-les index de la procédure (cf. INDEX),
-les variables locales (cf. VAR).

Les index sont une commodité qui permettent de prendre en compte la structure de tableaux. Vous avez le terreau pour utiliser les autres qui vous viendront à l'esprit: file, pile, list, arbre, monceau, etc..

VI. Structures algorithmiques:

Ne cherchez pas les boucles WHILE et FOR, construisez-les, c'est un très bon exercice, avec les instructions du tableau de la figure.3. Toutefois nous avons ici la notion de procédure et l'instruction d'insertion qui permettent de travailler et de construire ses propres fonctions. La procédure principale d'un programme sera la dernière procédure du fichier source et se termine par un point. Chacun des modules est formé de deux parties bien distinctes:

-la partie déclarative introduite par l'en-tête PROC, qui définit les données qui seront manipulées dans les corps de procédure,

-la partie exécutive qui commence par DEBUT et se termine par FIN. Elle contient les instructions exécutables qui représentent le traitement proprement dit.

VII. Exemples traités:

VII.1. Somme des n premiers entiers (ressources IBM PC):

Cet exemple, oh combien, simple dans son principe a pour but de donner un exercice qui permet de s'assurer du bon fonctionnement de l'instruction d'insertion. Le lecteur remarquera le traitement récursif du problème.

La somme des n premiers entiers est calculée de façon récursive. Ce qui constitue un exercice de style. La récursivité ne s'imposant pas à priori.

Nous donnons à la page suivante le listing du programme source et une exécution de l'exemple pour une valeur de n égale à 8. Le lecteur doit se rappeler que la procédure principale du programme est la dernière procédure du programme source.

PROGRAMME	CODE
	000 : 63 —) Adresse de la premiere configuration
	001 : 33 —) Adresse relative du programme principale 33+1 = 34
Proc Factoriel	002 : 00
Para 1	003 : 136 1
Index 0	005 : 144 0
Var 1;1	007 : 152 1 1
Debut	
Ecrire C(I)	010 : 130
Si C(3,0)=0 Vers Fin	011 : 8249 3 0 0 18
C(4,0):=C(3,0)-1	016 : 3097 4 0 3 0 1
Insere Factoriel(4)	022 : 112 65515 1 4
C(3,0):=C(3,0)+C(4,0)	026 : 2321 3 0 3 0 4 0
Fin : Fin	033 : 8
*	
* programme principal	
*	
Proc Main	034 : 00
Para 0	035 : 136 0
Index 0	037 : 144 0
Var 2;1,1	039 : 152 2 1 1
Debut	
Ecrire C(I)	043 : 130
Lire C(3.0)	044 : 121 3 0
C(4,0):=C(3,0)	047 : 2097 4 0 3 0
Insere Factoriel(4)	052 : 112 65485 1 4
Ecrire C(3.0)	056 : 129 3 0
Ecrire C(4,0)	059 : 129 4 0
Fin.	062 : 008

EXECUTION DE L'EXEMPLE No 1

Fichier a executer : ess1

-) 63	Adresse de la premiere configuration
? 8	valeur pour la somme recursive
-) 70	
-) 76	
-) 82	Evolution
-) 88	Des
-) 94	Configuration
-) 100	A
-) 106	Chaque
-) 112	Appel
-) 118	
-) 8	Rappel de la valeur
-) 36	Resultat 36 = 8+7+6+5+4+3+2+1

Ok

VII.2 Mini-dérouleur écrit en procédure formelle (~~IBM 3081-K~~): [KNIPPEL J.M. 89]

Cet exemple a pour but d'exploiter un code programme de procédure formelle que nous avons choisi. Le code programme est exécuté sur une configuration de données choisie par l'utilisateur. Les codes programme et de données sont figés dans le programme afin de simplifier la mise au point.

La compilation du code source est supposée faite dans une autre partie. Les instructions exploitées par le mini-dérouleur sont les suivantes:

code	type d'opérations	nombre d'opérandes
0	c(a):=c(b)+c(d)	3
1	c(a):=c(b)-c(d)	3
2	stop	0
3	vers ei	1
4	si c(a)=0 vers ei	2
5	c(a):=0	1
6	si c(w)=0 vers ei	1
7	c(a):=c(a)+1	1
8	c(a):=c(a)-1	1
9	c(a):=c(b)	2

Le code du source du programme traité dans le listing des pages suivantes est donné ici:

```
                                        c(3):=0                 5
3
BOUC:       si c(7)=0 vers stopons      4       7       11
                  c(7):=c(7)-1          8       7
                  c(3):=c(3)+c(6)       0       3       3       6
                  vers BOUC             3       65527
                  stopons:stop          2
```

L'utilisateur du programme donne seulement l'adresse de

début du code programme et du code des données.

PROGRAMME:

```
----------
        PROC CALCUL
        PARA 2
                INDEX 4
                        VAR 1;1
                DEBUT
                        C(5):-C(3)
                        C(11):-0
                        ENTREE:         SI C(5,0)-0     VERS PLUS
                                        SI C(5,0)-1     VERS PLUS
                                        SI C(5,0)-2     VERS STOP
                                        SI C(5,0)-3     VERS ALLERA
                                        SI C(5,0)-4     VERS COND
                                        SI C(5,0)-5     VERS ZERO
                                        SI C(5,0)-6     VERS SIOM
                                        SI C(5,0)-7     VERS PLUN
                                        SI C(5,0)-8     VERS MUN
                                        SI C(5,0)-9     VERS DPE
                         STOP:  SI C(11)-33 VERS ENDE
                                        ECRIRE C(3,0)
                                        C(3):-C(3)+1
                                        C(11):-C(11)+1
                                        VERS STOP
                                        COND:   C(6):-C(4)+C(5,1)
                                                SI C(6,0)-0 VERS COND1
                                                C(5):-C(5)+3
                                        VERS ENTREE
                                        PLUS:   C(6):-C(4)+C(5,1)
                                                C(7):-C(4)+C(5,2)
                                                C(8):-C(4)+C(5,3)
                                                SI C(5,0)-1 VERS MOINS
                                                        C(6,0):-C(7,0)+C(8,0)
                                                        C(9,0):-C(W)
                                                        C(5):-C(5)+4
                                                VERS ENTREE
                                                MOINS:  C(6,0):-C(7,0)-C(8,0)
                                                        C(9,0):-C(W)
                                                        C(5):-C(5)+4
                                                VERS ENTREE
                                                ALLERA          C(5):-C(5)+C(5,1)
                                                        VERS ENTREE
                                                        COND1:C(5):-C(5)+C(5,2)
                                                                VERS ENTREE

ZERO:C(6):-C(4)+C(5,1)
 C(6,0):-0
 C(5):-C(5)+2
VERS ENTREE
PLUN:   C(6):-C(4)+C(5,1)
        C(6,0):-C(6,0)+1
        C(9,0):-C(W)
C(5):-C(5)+2                                            VERS ENTREE
```

```
        MUN:    C(6):=C(4)+C(5,1)
                C(6):=C(6,0)-1
                C(9,0):=C(W)
                C(5):=C(5)+2
        VERS ENTREE
        SIOM:   SI C(9,0)=0 VERS SIOM1
                C(5):=C(5)+2
        VERS ENTREE
                SIOM1:C(5):=C(5)+C(5,1)
        VERS ENTREE
        DPE:    C(6):=C(4)+C(5,1)
                C(7):=C(4)+C(5,2)
                C(6,0):=C(7,0)
                C(5):=C(5)+3
        VERS ENTREE
ENDE:FIN
*
```

figure III .3. Source du mini-décalleur en P.F.

```
* PROGRAMME PRINCIPAL
        PROC MACHINEUNI
                PARA 0
                        INDEX 100
                        VAR 0
                DEBUT
                        C(6):=0 .....a1
                C(7):=0 .....a2
        C(8):=0 .....a3
C(9):=10        .....adresse du report
        C(10):=0        .....valeur de W
        C(11):=0        .....compteur pour l'édition des résultats
        C(12):=5        .....début du code programme compilé
                C(13):=3
                C(14):=4
                C(15):=7
                C(16):=11
                C(17):=8
                C(18):=7
                C(19):=0
                C(20):=3
                C(21):=3
                C(22):=6
                C(23):=3
                C(24):=65527
                C(25):=2        .....fin du code programme compilé
                C(26):=0        .....début  des DATA    EXECUTION DE L'EXEMPLE N°2
                C(27):=0                                ---------------------------
                C(28):=0                                  ?12....Adresse début du code
                C(29):=0        <--------------          ?26....Adresse début de DATA
                C(30):=0                                ->0
                C(31):=0                                ->0
                C(32):=3                                ->0
                C(33):=2        .....fin des DATA       ->6 <----------
        LIRE C(3),0)                                    ->0
        LIRE C(4),0)                                    ->0
        INSERE CALCUL(3,4)                              ->3
        FIN.                                            ->0
```

VIII. Fiche descriptive de l'outil:

MU-DE

non répertorié par ANL

Auteurs et diffuseurs:

Ch. Bisière, D. Iwanesko, J.L. Massat

E.Bianco, J.M. Knippel

Ressources: développé initialement avec la configuration IBM PC, le produit a été, à ce jour porté, sur:
MACINTOSH PLUS et ses évolutions,
IBM 3090, avec ressources logicielles VM/SP,
PASCALVS (Poste de travail: console alpha-numérique et Minitel, réseau: accès EARN).

Réalisation: auteurs: Ch.Bisière, D.Iwanesko, J.L.Massat.
début du projet: Juin 1986
volume: 1020 lignes de Pascal environ
état: prototype opérationnel aux Universités d'Aix-Marseille II et III depuis Janvier 1988.
langage utilisé: Procédure formelle.
dernière version: Avril 1989.

Diffusion: Pour tout renseignement s'adresser au LITAM.
Monsieur le Professeur E.Bianco.

Documentation:

Ch.Bisière, D.Iwanesko, J.L.Massat.
Projet de machine universelle. Rapport MMIAGe 85/87.

E.Bianco.Informatique fondamentale.Birkhauser Verlag
Bulletin d'informatique approfondie et applications.
Université d'Aix-Marseille II.N°0 à 22.ISSN0291-5413

Adresse: Université d'Aix-Marseille II
Faculté Pierre Puget
LITAM
14, rue Puvis de Chavannes
13001 MARSEILLE
Tél: 91 90 13 20

Mots clés:

Informatique fondamentale

MU-DE

Compilateur

Machine Universelle

non répertorié par l'ANL

Définition:

Compilateur de procédure formelle choisie pour ses critères pédagogiques d'enseignement et de formation à la recherche.

Decription fonctionnelle:

Le compilateur "MU" génère à partir d'un fichier texte source contenant un programme écrit en procédure formelle et les configurations de données nécessaires,un code qui sera ensuite exécuté grâce au dérouleur-exécuteur "DE" de procédu-

re formelle. L'utilisateur pourra ainsi se familiariser avec un langage qui permet à la fois une description aussi complète que possible des processeurs et ensuite de cheminer vers des notions de machines à cases adressables, de procédure formelle,de machine formelle, de compilation,de système de façon continue.....

.....

IX. Conclusion:

Nous avons pris le parti ici de présenter le compilateur de machine universelle à l'utilisateur, ce sont les paragraphes I et II. Nous ne sommes pas rentrés dans les phases explicatives concernant la décomposition du langage et l'analyse syntaxique, l'allocation-substitution, la génération de code. Le lecteur-connaisseur aura noté les modifications sur le langage de la procédure formelle "voyageuse" (7) dans le paragraphe III. Par la suite nous avons rappelé les types, et structures de données et de programmes qui seront à la base du développement des bibliothèques nécessaires: mathématiques, de manipulations de fichiers etc...

Enfin quelques exemples traités donneront le gôut au lecteur d'utiliser l'outil que nous décrivons synthétiquement au paragraphe précédent. Plus les critiques seront nombreuses plus nous pourrons aller vers une version que nous diffuserons par le réseau international EARN. Notre modestie devant en souffrir, les quelques deux cents étudiants de premier cycle et second cycle de nos Universités n'ont pas encore piégé les diverses implantations du produit actuel. Les développements sont à prendre en compte au niveau des types prédéfinis CHARACTER, pour qu'au delà du calcul arithmétique, la "parole" soit donnée au compilateur. Une crainte est de voir les étudiants "fuir le calcul" pour se réfugier dans "l'enluminure du calcul" et développer trop de procédures de saisie et d'édition repoussant la phase de réflexion du calcul.

A vous de suivre!

X. Bibliographie:
---

(1) E.BIANCO Le processeur expérimental MCA-0.
Bulletin d'informatique approfondie et applications. LITAM.
N°0. MARS 1981. ISSN 0291- 5413.

(2) E.BIANCO Le processeur expérimental MCA-0 (suite).
Bulletin d'informatique approfondie et applications. LITAM.
N°1. JUIN 1981. ISSN 0291-5413.

(3) J.Ph.LEHMANN Une preuve directe de l'équivalence entre machine de türing et machines à cases adressables.
Bulletin d'informatique approfondie et applications. LITAM.
N°6. DECEMBRE 1983. ISSN 0291-5413.

(4) E.BIANCO Notion de système. Système logiciel, système machine.
Bulletin d'informatique approfondie et applications. LITAM.
N°9. DECEMBRE 1984. ISSN 0291-5413.

(5) P.ISOARDI Une nouvelle machine...!
Bulletin d'informatique approfondie et applications. LITAM.
N°1. JUIN 1981. ISSN 0291-5413.

(6) E.BIANCO Notion d'insertion dans la machine de Nolin.
Bulletin d'informatique approfondie et applications. LITAM.
N°15. DECEMBRE 1986. ISSN 0291-5413.

(7) E.BIANCO Informatique fondamentale.
Birkhaüser Verlag. Bâle. 1979.

(8) Ch.BISIERE, D.IWANESKO,J.L.MASSAT.Projet de machine universelle
Rapport MMIAGe 85/87.

(9) R.J.CHEVANCE. Compilation. Notes de Cours.
Institut de Programmation. Université de Paris VI. 1984.

(10) EBIANCO Notion d'autojectivité. Notion de statut.
Bulletin d'informatique approfondie et applications. LITAM.
N°19. MARS 1988. ISSN 0291-5413.

VOUZZAVEDIBISAR

Il y a des concepts, comme ça, qui vous amèneraient à douter de la cohérence de la pensée moderne, aussi les assertions suivantes ne sont elles pas à mettre entre toutes les mains.

_ Dieu a tous les pouvoirs, il a donc obligatoirement celui de s'annihiler lui-même.

_ La Liberté c'est comme la mer, ça vous arrive par vagues. Quand la vague déferle elle vous noie, quand elle est passée vous avez le temps de vous dessécher sur le sable. La Liberté c'est aussi être libre d'entraver la liberté des autres.

_ L'Honneur appartient aux gens qui ont l'habitude de faire leur petite lessive avec le sang des autres. L'Honneur lave plus rouge.

Amour informatique et virus

_ Souffrez Marquise que je vous offre mon Amour par la grâce de cette disquette...

_ Ah... Prince que vous me comblez... Oserai-je vous suggérer de revêtir votre friponne disquette d'un délicat préservatif ?