Serveur de dialogue multi-activités, multi-fenêtres et multi-consoles

Lucien Rorato

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE AGREE
DE GRENOBLE (C.U.E.F.A)

MEMOIRE

présenté en vue d'obtenir

le diplôme d'ingénieur

en

Informatique

par

Lucien RORATO

SERVEUR DE DIALOGUE
MULTI-ACTIVITES, MULTI-FENETRES ET MULTI-CONSOLES

SOUTENU LE : *13 Février 1984*

JURY

Président : *Professeur J. V. RANCHIN*

Membres : *Professeur L. BOLLIET*
JF. LEMERRE
Ch. TRIOLAIRE

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE AGREE
DE GRENOBLE (C.U.E.F.A)

MEMOIRE

présenté en vue d'obtenir

le diplôme d'ingénieur

en

Informatique

par

Lucien RORATO

SERVEUR DE DIALOGUE
MULTI-ACTIVITES, MULTI-FENETRES ET MULTI-CONSOLES

SOUTENU LE : *13 Février 1984*

JURY

Président : *Professeur J. V. RANCHIN*

Membres : *Professeur L. BOLLIET*
JF. LEMERRE
Ch. TRIOLAIRE

Le travail présenté dans ce mémoire a été entrepris à la SEMS (Société Européenne de Mini-Informatique et de Système, devenue BULL-SEMS depuis) dans le cadre de la Formation Permanente.

J'adresse mes remerciements

à Monsieur le Professeur RANCHIN, du Conservatoire National des Arts et Métiers, qui m'a fait l'honneur de présider le jury.

à Monsieur le Professeur BOLLIET, de l'Université Grenoble II, qui m'a fait l'honneur d'accepter le dossier de thèse.

à Monsieur J.F. LUMINEAU, Chef de Division, qui m'a permis de réaliser ma thèse dans son département ;

à Monsieur Ch. TRIOLAIRE, Chef de Service, pour l'intérêt qu'il a témoigné en suivant ma thèse durant toute sa durée ;

à Monsieur P. MILESI pour les conseils qu'il m'a prodigués ;

à Messieurs J.C. CHUPIN, Directeur Technique, et J.F. LEMERRE, Chef de Service qui m'ont permis de terminer ma thèse dans de bonnes conditions.

- TABLE DES MATIERES -

1 INTRODUCTION

Le sujet de ma thèse était d'étudier comment la console 'télévidéo 950', nouvelle acquisition de la SEMS, pouvait contribuer à améliorer la qualité du dialogue homme-machine sur le matériel de la gamme SOLAR.

Cette étude a abouti à la réalisation d'un serveur de dialogue. SEDRIC, SErveur de Dialogue à Fonctions Répartissables et Indépendant Console, est destiné à libérer le programmeur des contraintes inhérentes au terminal. A travers un jeu de primitives simples, il fournit, dans ce but, une interface standardisée de communication vers la console.
Il facilite et agrémente la tâche de l'opérateur en lui offrant un dialogue à la fois riche et souple.

Proposant, tout d'abord, la définition de certains termes, nous apporterons ensuite quelques éléments de réflexion sur le dialogue homme-machine.

Nous exposerons, alors, le serveur de dialogue lui-même. A la présentation des buts fixés, suivra celle de l'interface logicielle puis opérateur.

Succèdera alors la description de la réalisation.
Commençant par le logiciel de base utilisé, nous continuerons par l'une des fonctionnalités importantes de SEDRIC, viendront alors l'architecture, les divers mécanismes mis en oeuvre.
Nous analyserons, ensemble, les primitives et l'interface usager.
Le dernier chapitre sera réservé à certains éléments que nous aurions aimé adjoindre à SEDRIC.
C'est alors que nous conclurons notre exposé.

2 TERMINOLOGIE

Opérateur
: Personne travaillant sur la console.

Module Client
: Tout module, programme, tâche ou processeur demandant les services de SEDRIC.

Segment d'Ecran
: Portion de l'écran physique constitué d'une suite de lignes entières et contigues. Un segment est nommé. L'écran physique peut supporter plusieurs segments qui peuvent se recouvrir.

Segment Actif
: Segment sur lequel l'opérateur peut agir (plusieurs segments peuvent être actifs).

Segment courant
: Segment sur lequel l'opérateur est en train d'agir (tout segment actif peut devenir courant grace à une sélection).

Ecran virtuel
: Mémoire associée au segment d'écran. Il sert de moyen de communication entre le module client et SEDRIC.

Ecran virtuel courant ou actif
: Celui associé au segment courant ou actif.

Fenêtre virtuelle
: Portion de l'écran virtuel affichée sur le segment d'écran.

Fenêtre virtuelle courante ou active.
: Fenêtre affichée sur le segment courant ou actif.

Multi-fenêtres
: Propriété donnée à un terminal pour permettre de découper l'écran en plusieurs fenêtres.

Multi-activités
: Un dialogue est dit 'multi-activités' s'il permet de travailler, sur un même écran, avec des modules clients différents à l'intérieur de fenêtres différentes.

Menu
: C'est une liste d'ITEM numérotés, présentant des choix à l'opérateur qui, à l'aide du numéro, en sélectionne un parmi ceux affichés. Ce choix produit une action telle que :

- affichage d'un autre menu, appelé menu fils,
- affichage d'une grille demandant à l'opérateur de fournir des paramètres,
- transmission de la réponse au module client. Celle-ci peut être :
  - un choix,
  - une suite de choix,
  - un choix ou une suite de choix suivis de paramètres.

Les menus ont donc une structure arborescente.

Historique

Associé à un écran virtuel, il n'existe que si le dialogue s'est effectué avec des menus. Il en retrace la dernière arborescence en relatant l'ensemble des choix pris, ainsi que les paramètres fournis. C'est, lui-même, un menu.

Rafraichissement

Mise à jour d'un segment non actif à la suite d'une demande d'affichage d'un module client.

Message

Suite structurée d'informations utilisées pour la communication entre les modules clients et le serveur de dialogue ainsi qu'entre les divers modules de SEDRIC.

Voici un schéma fonctionnel des différentes entités que nous venons de définir.

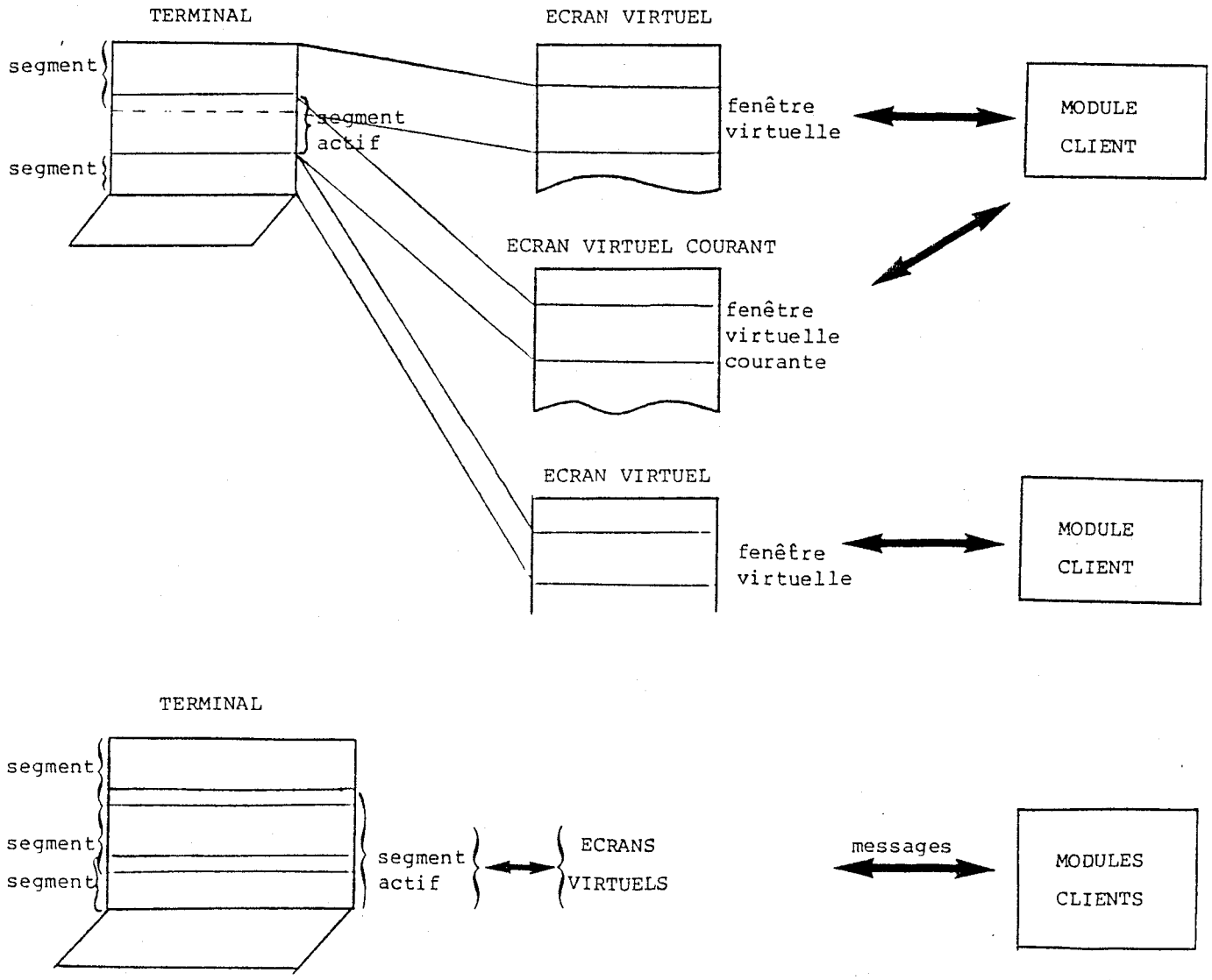


Fig. 1

3 LE DIALOGUE HOMME-MACHINE

3.1 GENERALITES

Ce chapitre présente quelques éléments de réflexion sur les critères de qualité d'un dialogue homme-machine. Aujourd'hui, celui-ci contribue, de plus en plus, à mesurer la qualité d'un produit informatique. C'est devenu pour un constructeur un atoût de vente important. Lors de la conception d'un produit il faut, désormais, s'attacher à considérer cet aspect dès l'origine.

La démarche que nous proposons consiste à déterminer, pour chaque produit à concevoir, en fonction des objectifs à atteindre, l'importance que l'on souhaite donner à tel ou tel aspect de la communication avec la personne. Pour permettre de suivre cette démarche nous avons identifié des critères de qualité.

Dès à présent, insistons sur ce point : certains critères peuvent parfois sembler s'opposer entre eux. Ces oppositions se présentant en des termes différents suivant les applications. Et c'est au concepteur qu'il appartient de décider de l'importance à donner à tel ou tel critère. C'est lui qui détermine, en fonction des objectifs poursuivis et des contraintes d'utilisation et de réalisation, le profil que devra présenter le produit par rapport à l'ensemble de ces critères.

Un schéma, tel que celui présenté ci-dessous, devrait pouvoir être établi en songeant que le nombre d'étoiles mesure le niveau que le concepteur voudra atteindre pour le critère correspondant.

Critères	Produit-P1	Produit-P2
A	***	*
B	*	**
C	****	**
D	**	*
E	**	****

Soucieux de clarifier la présentation, nous avons choisi de classer les différents critères en familles. Ce sont par ordre alphabétique :

Adaptable
Confortable
Efficace
Informatif
Pédagogique
Rétrospectif
Tolérant

En voici maintenant la présentation.

3.2 ADAPTABLE

Quatre critères donnent, selon nous, au dialogue la qualité d'adaptable. Ce sont :

L'adaptation au niveau de connaissance

L'adaptation au langage

La "docilité"

L'adaptation par profil.

Adaptation au niveau de connaissance

Entre le novice, l'usager occasionnel et l'expert existent, en effet des différences de comportement et de besoins. Pour résoudre ce problème le choix du type de communication peut porter sur :

- Le mode : du guidé au non guidé
- Le type de langage : pseudo-naturel; plus compact, utilisation de menus et d'aides de type 'Help'.
- Les messages : de plusieurs niveaux, du concis à l'explicatif.

Adaptation au langage

Selon l'application, selon la profession de l'usager ou selon ses habitudes linguistiques, le langage utilisé peut différer. La faculté d'adaptation correspondant concerne :

- Le texte des messages.
- La syntaxe utilisée pour les commandes.
- et plus généralement le langage.

On peut réaliser cela par redéfinition de commandes, de mot-clés et d'éléments de lexique.

Docilité

La communication doit être telle que l'usager puisse garder l'initiative, et, à tout moment obtenir des changements de comportement, tels que :

- changement du mode de communication (menu, langage de commande);
- choix du niveau de message;
- interruption du dialogue (arrêt d'un bavardage excessif);
- reprise d'un dialogue interrompu;
- arrêt d'une action trop longue ou paraissant suspecte.

Adaptation par profil

Le profil de "communication" fait référence au niveau de connaissance, à la méthode de dialogue, au contenu des messages, à la langue. Il doit être considéré comme distinct du profil "système" qui lui définit les droits d'accès et les domaines d'utilisation liés à une application pour des impératifs de sûreté ou d'efficacité. Ces deux profils peuvent cependant

être associés.

Le profil de communication peut être typé en fonction de l'appartenance à un groupe. Il définit le mode de communication à utiliser au temps initial. Il est modifiable dynamiquement, sur initiative de l'usager; les modifications admissibles dépendent elles-même, du type de profil.

Il se conçoit également comme un profil controlé par le système, défini par celui-ci à la suite d'observations sur le comportement de l'usager. On peut à partir d'un ensemble d'actions choisies au cours d'un dialogue guidé, proposer de désigner celle-ci par un nom ou de les résumer par un menu. Lesquels pourront être ultérieurement utilisés.

3.3 CONFORTABLE

Voici un qualificatif qui, appliqué à un dialogue informatique, peut sembler surprenant. On veut dire, par ce terme, que la communication doit tenir compte des critères ergonomiques propres à assurer une meilleure efficacité dans le travail.

On constatera que nous nous sommes plus attachés à étudier les éléments de confort psychologiques, laissant aux spécialistes en ergonomie le soin d'étudier les aspects physiologiques.

Nous avons retenu sous cette rubrique quatre critères qui sont :

- un organe de dialogue ergonomique;
- l'homogénéité;
- la brièveté du temps de réponse;
- un dialogue communicatif;

En voici l'explication.

Organe de dialogue ergonomique

La présentation à l'opérateur doit être agréable et efficace. Elle privilégie l'image et le son sur le texte et s'efforce de montrer un dessin plutot que d'imprimer une description.
De même, on préfèrera l'action qui consiste à désigner un objet directement à celle qui est de déplacer un repère et d'appuyer sur une touche. Néanmoins, dans un souci de précision (C.A.O.) elle peut être plus intéressante que la précédente. Elle sera toujours préférable à la frappe de directives sur un clavier.
L'usager désire de plus en plus que l'organe de dialogue dont il dispose lui permette de recréer l'univers de sa table de travail (fenêtres multiples, correspondant à la diversité des documents). Il souhaite être aidé dans le domaine de la création des textes et des polices de caractères. On pourra avantageusement, pour cela, prendre connaissance des réalisations telles que SMALLTALK 80, LISA, PERQ, APOLLO et STAR.

Homogénéité

Il est bon que le dialogue soit homogène entre deux applications afin que l'opérateur ne soit pas perturbé ou désorienté par des changements inutiles. Si les objets présentés le sont sous la même forme il n'y aura ni perte de temps, ni fatigue supplémentaire dans un effort de familiarisation. Il en est de même pour la manière de soumettre une requête et de prendre connaissance d'un message.

Brièveté du temps de réponse

Elle ne dépend pas essentiellement du logiciel et des périphériques dédiés au dialogue mais bien souvent de la qualité du système, de l'application et du matériel les supportant.
Certaines techniques, telle l'anticipation dans l'analyse des réponses de l'opérateur, contribuent à diminuer ce temps de réponse. (Ex: CORAIL). Dans tous les cas, pour être bien acceptés il doit être en cohérence avec le rythme humain. D'une durée constante, si possible, il doit être conforme au temps d'attente escompté. Ainsi il ne provoque pas d'impatience.

Communicatif

On pourrait aussi dire médiatif. Seul, face à son terminal, isolé, le plus souvent, l'usager doit pouvoir trouver une contrepartie à cette situation en disposant de moyens de connaissance de la communauté des utilisateurs. Il souhaite savoir qui est présent, avec qui il peut communiquer pour obtenir une information sur une aide ou transmettre un message ou un avertissement. Il souhaite communiquer aussi avec certains, temporairement absents.

3.4 EFFICACE

La communication par le dialogue doit favoriser l'efficacité de la personne qui utilise l'outil informatique en lui permettant de résoudre les problèmes. Pour cela, éviter ce qui freinerait son efficacité naturelle.

Nous avons retenu quatre critères dans cette famille.

Non dispersion

Il s'agit d'éviter tout ce qui n'est pas utile à la résolution directe du problème : ne donner à l'usager que ce qui l'intéresse en lui fournissant seules les informations directement associées à ses actions et en ne lui faisant pas obligation d'emprunter des cheminements qui ne sont pas directement utiles.

Automaticité

Permettre la redéfinition et l'association de demandes (exemple : procédures de commandes) pour permettre d'éviter la répétition de séquences similaires. Egalement evoquées sous la rubrique adaptabilité, ces séquences d'exécution, doivent pouvoir se réaliser de façon personnalisée par paramétrisation et dialogue.

Ubiquité

Le don d'ubiquité : "être à plusieurs endroits à la fois", consiste ici à permettre à l'usager un changement facile d'environnement.
Il s'agit d'éviter d'enfermer l'usager dans le mode de fonctionnement qu'il a obtenu et de supprimer la frontière entre les modes :

- possibilité de garder un environnement tout en allant travailler dans un autre (ex : demande d'informations système pendant la modification d'un objet).

- Activation multiple de plusieurs travaux. Ils peuvent simultanément présenter des résultats ou fournir des informations dans des fenêtres-écran distinctes sur un même écran physique.

- Visualisation simultanée de plusieurs dialogues.

Remarque :

L'implémentation de ce dernier critère est appelée "multi-activité". On le trouve notamment sur 'APOLLO'. Nous l'avons réalisé dans SEDRIC.

3.5 INFORMATIF

L'usager attend de son interlocuteur-système qu'il l'informe sur ses actions présentes ou passées, qu'il tienne à sa disposition, des informations sur ce qui concerne le système et son activité, en évitant de tomber dans l'excès de bavardage.

Le système sait informer sur :

Ce qu'il a compris

Les objets qu'il utilise

Les actions qu'il réalise

L'état d'avancement des travaux en cours.

Ces informations peuvent être fournies de façon systématique ou à la demande.

Le système peut signaler les conséquences directes ou indirectes d'un travail demandé et informer sur sa charge actuelle si elle peut avoir des conséquences pour l'usager. Eventuellement il fournit un devis estimatif en delai pour la tâche envisagée.

L'usager doit être également informé à tout moment sur son contexte de travail. Il a besoin de savoir d'où il vient, où il est, ce qu'il peut faire à cet endroit, où il peut aller et comment il peut exprimer ses demandes.

Ce critère ne doit pas entrainer de surabondance d'information. Elle ne devra être délivrée qu'à bon escient, pour celui qui en a besoin, et en qualité limitée à chaque fois. Mieux valent de petits lots réclamés à la demande qu'un gros paquet long à présenter et difficile à assimiler.

3.6 PEDAGOGIQUE

L'outil informatique peut apporter à l'usager des éléments pédagogiques qui lui permettent de se former à une meilleure utilisation. Il est souhaitable de pouvoir travailler avec le terminal et les fonctions disponibles sans avoir auparavant appris à les utiliser.

Ces critères correspondent à un besoin qu'on doit exprimer sous 2 aspects :

- Le système se suffit à lui-même; il n'est pas nécessaire de posséder une documentation imprimée.

- l'usager doit pouvoir trouver les moyens de son propre apprentissage. Cet apprentissage doit pouvoir se réaliser de façon simple, sans douleur, et au rythme de l'usager. Exemple : LEARN sur UNIX.

Ce besoin n'est pas ressenti par le seul novice; il est aussi ressenti par l'usager occasionnel même chevronné qui peut oublier des détails. De toute façon, les besoins peuvent évoluer et un complément de connaissances est souvent nécessaire.
Ces aspects pourront être satisfaits par les caractères suivants :

Auto explicatif

L'action sur le système doit être obtenue par l'émission de demandes dont le nom est clair et la syntaxe compréhensible, tout en restant simples et concises. De même les messages doivent être émis le plus clairement possible, sans "verbosité" excessive. Mieux vaut une réponse sous forme d'un mot compréhensible, qu'une abréviation; mieux vaut une abréviation qu'un numéro.

Explicatif à la demande

L'usager en difficulté doit pouvoir exprimer qu'il a besoin d'une aide pour comprendre une réponse ou obtenir une explication sur une situation qu'il ne comprend pas. Il doit pouvoir accéder à des fonctions d'aide de type "HELP". Celles-ci peuvent être accessibles à plusieurs niveaux, avec des degrés d'explication progressifs. De plus, Il est essentiel que soient fournies à la demande les coordonnées de la personne physique (administrateur ou spécialiste), qui pourra donner un conseil ou fournir une aide matérielle.

Documentation accessible

L'information pédagogique et la documentation sous forme de brochures ou les consignes d'utilisation sont difficilement accessibles. Il est agréable de pouvoir les obtenir directement sur le poste de travail. Le fait que le document soit en ligne, accessible au bout des doigts, pourra faire gagner un temps considérable à l'usager isolé ou non pourvu de documents.

Auto éducatif

L'usager novice doit pouvoir pressentir, à travers le dialogue qui lui est dédié, l'existence d'un mode opératoire différent et plus performant. Ainsi l'apprentissage pourra se faire sans beaucoup d'efforts.

Cohérence entre niveaux

Quand divers niveaux de dialogue sont accessibles pour un même usager, la cohérence entre ces niveaux doit exister. Les termes et la syntaxe utilisés lors d'un dialogue guidé préfigurent ceux qui seront nécessaires lors d'un dialogue non guidé; ceci pour permettre l'apprentissage lors d'une évolution vers un niveau de dialogue moins assisté.

3.7 RETROSPECTIF

Pour certains types d'application, il est souhaitable de mettre au service de l'usager le pouvoir de mémorisation du système. Il est, en effet, parfois utile d'avoir une vue rétrospective sur les actions réalisées et de pouvoir revoir l'histoire, avec utilisation éventuelle d'un filtre pour une sélection selon des critères tels que la date, le type d'action ou l'objet traité. Certains éléments peuvent en effet être nécessaires pour permettre de :

- se souvenir : élément de confort, mais aussi élément nécessaire pour prendre une décision.

- reconstituer l'historique, dans un but de statistiques ou de comptabilité.

- définir des éléments de réalisation d'un profil utilisateur ou d'une séquence de demandes.

A cette notion de rétrospectivité, vient se greffer celle de rétroactivité, également envisagée sous le critère de la tolérance. Il ajoute au pouvoir de se souvenir, le pouvoir de défaire et de revenir à un état antérieur, quand cela est envisageable.

Rétrospectivité et rétroactivité sont combinés dans SEDRIC sous la forme de l'historique.
L'écran virtuel, quant à lui, agrémenté, des fonctions "window up' et 'window down' est un exemple de rétrospectivité simple.

3.8 TOLERANT

La tolérance s'exprime par l'acceptation du fait que l'être humain fait des erreurs. Il est fondamental qu'une erreur de dialogue ne soit pas irrémédiable.

Un autre aspect est la difficulté de communication : pour bien communiquer, il est nécessaire que les partenaires fassent chacun un effort de compréhension de ce qu'a voulu dire l'autre.
Nous avons regroupé quatre critères dans cette famille.

Concision

La concision du dialogue permet de demander moins d'effort à l'usager. De plus, elle diminue les risques d'erreur. Elle se concrétise par :

- l'utilisation d'abréviations. Celles-ci sont admises dans la mesure où elles n'introduisent pas d'ambiguité.

- la suppression des informations inutiles ou superflues.

Correction automatique

Quand des erreurs de syntaxe sont identifiées, elles peuvent être parfois corrigées automatiquement, en prenant toutefois des précautions et en informant, bien sûr, l'usager de son erreur et du choix fait.

Mise en garde et demande de confirmation

Quand le dialogue induit des actions importantes ou déterminantes pour la suite il est parfois nécessaire d'entamer un dialogue particulier; celui-ci a pour but d'informer le demandeur des conséquences probables de sa demande et de lui en réclamer confirmation. Ce processus peut s'engager sur présomption d'une faute de dialogue ou pour simple raison de sécurité. La forme que prend ce dialogue dépend du degré de gravité; selon que l'action entraine une perte de temps, une action manuelle, une destruction etc.

Annulation d'action

Quand cela est envisageable, il est bon de fournir la possibilité d'annuler la dernière action demandée (ou les dernières). Le fait de pouvoir revenir en arrière et défaire ce qui a été fait peut être déterminant pour encourager l'esprit d'initiative et d'audace chez le débutant. C'est surtout un facteur de sécurité non négligeable.

Cette fonctionnalité est offerte à travers la commande 'UNDO' sur les postes de travail que sont 'LISA' et 'SMALLTALK' entre autres.

4 PRESENTATION DE SEDRIC

4.1 LES BUTS FIXES

Ce sont :

- une bonne opérabilité,
- une indépendance vis-à-vis du terminal,
- une facilité du départ des fontions.

4.1.1 Opérabilité

Plusieurs éléments dans SEDRIC contribuent à atteindre cet objectif.

A) Multi-fenêtre
L'opérateur éprouve souvent le besoin de garder présent sous les yeux ou de consulter instantanément des informations provenant de différents modules clients. Effectuant un travail de facturation, il voudra conserver dans un coin de l'écran la liste des tarifs pour les produits concernés. C'est pourquoi nous avons choisi de découper l'écran en segments ou fenêtres physiques.

B) Multi-activité
Cette fonctionnalité, complément de la précédente, donne à l'utilisateur la possibilité de faire dérouler sur le même écran mais dans des segments différents plusieurs modules clients simultanément. L'affichage perpétuel d'information système peut se faire en même temps qu'une opération de production de programme. Multi-activité et multi-fenêtre contribuent à donner l'impression à l'opérateur de disposer de plusieurs écrans sur le même terminal.

C) L'Ecran virtuel (E.V.).
En mémorisant des informations passées mais pas forcément périmées il présente un double avantage. D'une part, il permet de s'affranchir de la taille de l'écran et des fenêtres physiques. D'autre part, il tend à supprimer ce caractère fugitif qui est le propre des informations affichées sur une console. L'opérateur peut ainsi consulter une phase antérieure du dialogue afin de répondre à une sollicitation présente.

D) Mode d'utilisation du terminal
On adopte un mode d'utilisation (mode bloc) dans lequel on dispose de toutes les fonctions d'édition du terminal (insertion, suppression de caractères, etc...). La construction du texte en est facilité sans aggraver le temps de réponse.

E) Menu
Le dialogue est souvent rébarbatif de part la syntaxe de ses commandes. L'utilisation de menus pallie à cet inconvénient. De plus, il guide l'opérateur dans son choix. Pour le compléter l'affichage d'une grille lui indique les paramètres à préciser. Les risques d'erreurs de syntaxe sont ainsi minimisés. Cela conduit également à une uniformisation du dialogue.

F) Historique
Attaché à l'utilisation de menus, cette fonctionnalité mémorise les

différents items choisis tout au long du dialogue. Elle permet de constituer un profil que l'on peut consulter pour le 'rejouer' en partie ou en totalité. Evitant ainsi de dérouler plusieurs fois des menus pour lesquels on fournit systématiquement les mêmes réponses.

F) Transparence aux commandes
Le système de menus peut paraître lourd à un opérateur expérimenté. C'est alors un dialogue trop prolixe. Pour cette raison, il est tout à fait facultatif. Nous autorisons, en effet, l'utilisation d'un langage de commande propre au module client et pour lequel SEDRIC est tout à fait transparent.

G) Modularité du dialogue
La multiplicité des fenêtres physiques permet de séparer les différents composants d'un dialogue pour lui donner plus de rigueur.
A la demande du module client ou de l'opérateur, SEDRIC peut réserver un segment d'écran, par exemple, pour :

- les menus et l'historique,
- le texte élaboré lors d'une édition,
- les commandes,
- le menu de sélection des segments.

Sans, pour autant, cloisonner l'utilisateur dans un mode de fonctionnement particulier.

Généralisant cette notion nous permettons à un même module client d'utiliser plusieurs fenêtres physiques.
En phase de mise au point, on peut imaginer qu'un module client ait l'usage de trois segments :

- l'un pour le titre,
- l'autre pour l'affichage et la saisie courante,
- le dernier pour la visualisation d'une trace.

4.1.2 Indépendance vis-à-vis des terminaux

Sur un même site, on est souvent contraint de disposer de consoles différentes. Cela pour des raisons de coût, de spécificité des applications, ou autres.
De même, poussé par les impératifs du marché, un constructeur se doit, bien souvent d'offrir des terminaux de visualisation de types divers.
Il en est de même avec l'évolution de la technologue. Le parc des périphériques (entre autre) est réactualisé périodiquement.
Tous ces facteurs ne sont pas sans conséquence sur le logiciel. Si l'hétérogénéité des terminaux n'a pas été prévue dès le départ, elle peut alors être d'un coût important.
C'est pourquoi, nous avons conçu un module dédié aux échanges avec la console. Il a été réalisé de telle sorte que les modifications à apporter en cas de changement de périphérique soient aisées et minimes.

4.1.3 Déport des fonctions

Cela permet d'utiliser l'intelligence des terminaux en leur faisant exécuter certaines tâches. Celles-ci sont d'autant plus nombreuses que le terminal est sophistiqué.
L'architecture en couche que nous avons adoptéeen facilite le transport. Chaque couche, conçue de façon modulaire est spécialisée dans une fonction.
Ce déport soulage le calculateur hôte et contribue à un gain de performance sur l'ensemble de l'application.

Chronologiquement, on peut voir le déport des fonctions de la manière suivante :

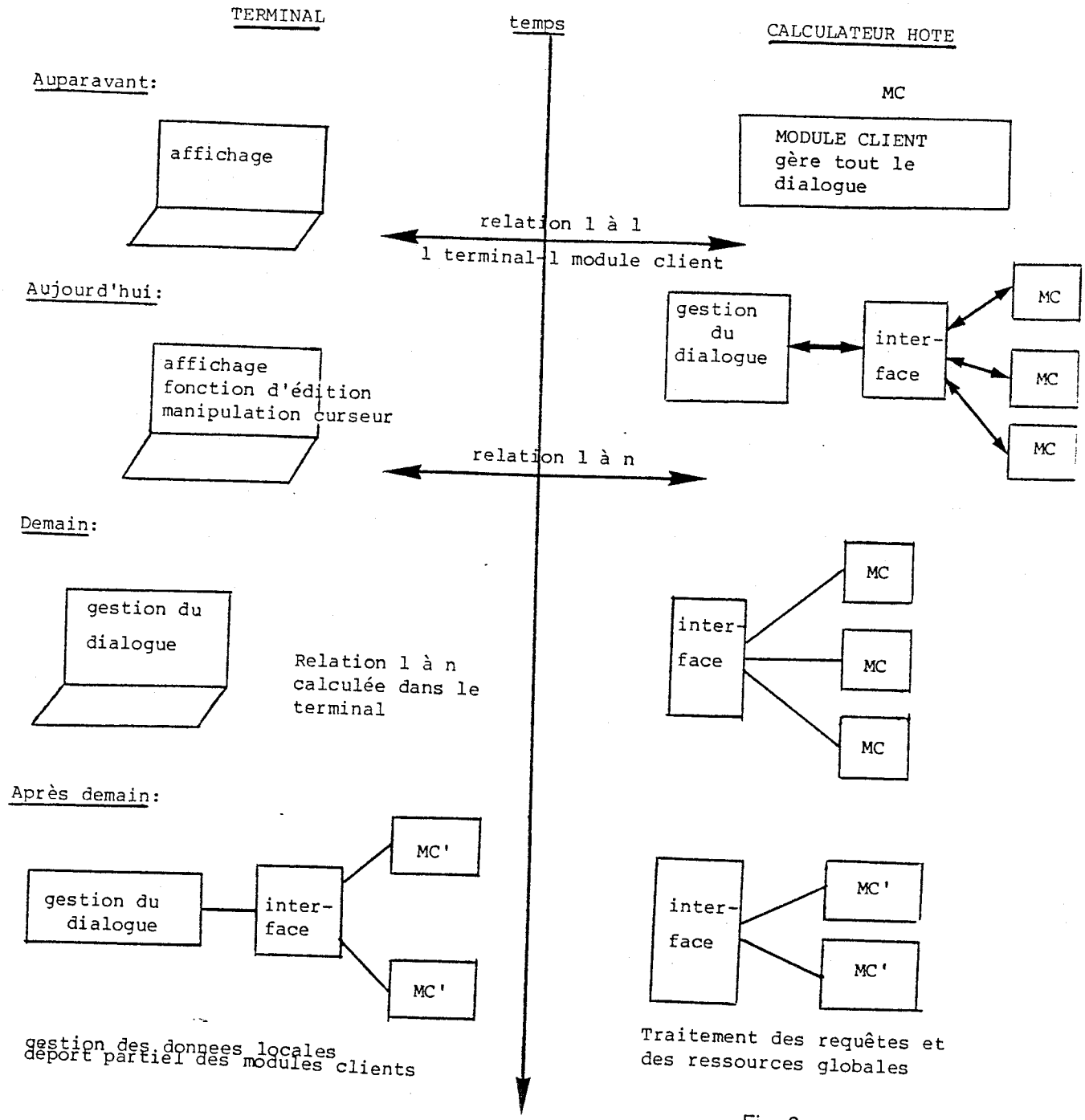


Fig. 2

4.2 L'INTERFACE LOGICIEL

4.2.1 Généralités

Nous nous sommes efforcés d'offrir une interface d'utilisation aisée. Nous avons, ainsi, volontairement restreint le nombre des primitives. L'appel de SEDRIC se fait à l'aide d'un sous-programme de la bibliothèque spécifique de SEDRIC. Il est le même quelque soit la primitive désirée. Il y a toujours deux paramètres qui sont l'un, le message d'appel; l'autre le message de compte-rendu. Le type de la primitive et tous les paramètres sont fournis dans ces messages. De ce fait, ils sont pour la plupart facultatifs et leur ordre d'apparition est nullement imposé. La signification du paramètre ne dépend pas de sa position mais du sélecteur qui le précède.
Nous allons maintenant présenter chacune de ces primitives. La description de leur implémentation respective fera l'objet d'un chapitre spécifique.

4.2.2 CREATVS (Creat Virtual Screen)

Cette primitive permet au module client d'amorcer la communication avec le terminal et d'initialiser le dialogue.
Il lui est alors affecté

- un écran virtuel,
- une fenêtre virtuelle et
- un segment d'écran.

Grâce à cette primitive, il peut faire connaître à SEDRIC le nom du premier menu à afficher (racine de l'arbre des menus).

Voici comment se décompose le message d'appel :

01	Type de la primitive
05	Sélecteur du nom de l'écran virtuel
VSNAME	Nom de l'écran virtuel
06	Sélecteur de la taille de l'écran virtuel
VSSIZE	Valeur de la taille
07	Sélecteur de la taille du segment
SEGZISE	Taille du segment
08	Sélecteur du nom du menu
MENUNAME	Nom du menu
09	Sélecteur des délimiteurs
Delimiters	Suite de délimiteurs
10	validation du dialogue par commande, tout en se réservant la possibilité d'utiliser les menus.
11	Sélecteur du numéro de périphérique
DFVICENUM	Numéro identifiant le périphérique
05	On peut après celà décrire un autre
VSNAME	écran virtuel si le module client en utilise plusieurs. Cette description peut se répéter autant de fois qu'il y a d'écrans virtuels (hormis les délimiteurs et le numéro du terminal).
.	
.	
.	
CODE-EOT	

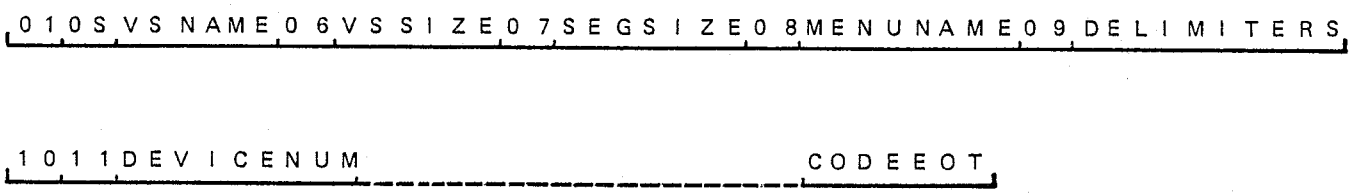


Fig. 3

4.2.2.1 Effet au niveau du dialogue

Il dépend de la présence ou de l'absence de menu.
Dans le second cas, il y a seulement matérialisation de la fenêtre physique sur l'écran par un cadre.
En plus de cela, on affiche, dans le premier cas, le menu spécifié. On active, par la suite, ceux induits par les choix de l'opérateur. Le contenu du message de retour relate l'ensemble de ces choix.

Le message de compte-rendu est le suivant :

A) Avec menu

01	Indique un compte-rendu de CREATVS
CRD	Valeur du compte-rendu
VSNUMBER	Numéro de l'écran virtuel
32	Signale une réponse à un menu
NBCHOIX	Nombre de choix
CHOIX1	Choix effectué pour le premier menu
.	
.	
.	
CHOIXn	Choix terminal
LGP1,	Nombre de caractères pour le premier paramètre
P1,	texte du premier paramètre
.	
.	
.	
LGPn,	
Pn	
CODE-EOT	

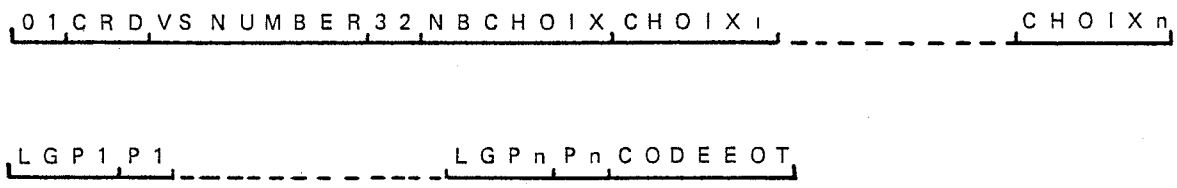


Fig. 4

B) Sans menu

01	Compte-rendu de CREATVS
CRD	Valeur du compte rendu
VSNUMBER	Numéro de l'écran virtuel
CODE-EOT	indicateur de fin de message.

Compte-rendu d'erreur :

Il peut prendre les valeurs suivantes :

0	: pas d'erreur
1	: erreur de syntaxe
2	: l'écran virtuel est surdimensionné ou existe déjà
3	: le terminal est hors-service

4.2.2.2 Remarques :

1) Nom de l'écran virtuel :
Le nom fourni pour les autres primitives est celui de la primitive creat concaténée avec le numéro donné en compte-rendu. Ceci permet de différencier les écrans virtuels créés par un même module client et avec le même nom. De plus, cela permet d'assurer l'intégrité des écrans virtuels.
Le nom d'un écran virtuel comporte au maximum huit caractères. Il est obligatoire.

2) Taille de l'écran virtuel
Elle est de 50 lignes par défaut. C'est également la taille maximale pour la maquette réalisée sur SOLAR.

3) Taille du segment
Au maximum de 24 lignes elle en fait 10 par défaut.

4) Nom du menu
Comme les deux paramètres précédents il est facultatif. Il est formé au plus de huit caractères.

5) Les délimiteurs
Au nombre de quatre, ils sont utilisés dans les messages structurés en champs. Ainsi, pour la primitive 'DISP' (affichage d'un texte), ils servent à structurer le message d'appel. Leur redéfinition ne s'impose que si les valeurs prises par défaut (32768 + SOH, 32768 + STX, 32768 + ETX, 32768 + EOT) ne conviennent pas. Ces valeurs doivent être supérieures à 32768.
L'ordre de redéfinition est imposé. C'est le suivant :

1) délimiteur de descripteur de texte,
2) de début de texte,
3) de fin de texte,
4) de fin de message.

4.2.3 DISP (DISPLAY)

Permet d'afficher sur le segment d'écran :
Un texte,
Ou, un buffer formatté.

Un texte étant, ici, considéré comme une suite informelle de caractères.

Un buffer formatté étant, lui, une suite de caractères structurée. Il est encore dénommé "grille".

Cela se traduit sur l'écran par l'affichage de caractères regroupés en zone dont certaines sont dites protégées (l'opérateur ne peut pas en altérer le contenu); d'autres sont modifiables. C'est à l'intérieur de ces dernières que l'usager fournit les réponses attendues par le module client. Les zones sont également appelées blocs.

La structure du message d'appel est la suivante :

02	Indique le type de la primitive (DISPLAY)
VSNAME	
VSNUMBER	Numéro de l'écran virtuel
type-traitement-initial (TTI)	Sert, par exemple, à demander l' effacement préalable d'un segment d'écran.
CODE-SOH	Indique le début de la description d'une zone.
TEXTSIZE	Nombre de caractères formant le texte. Ce paramètre obligatoire doit toujours se situer derrière 'CODE-SOH'.
18	Code indiquant un champ positionnement
LIGNUMGER	Numéro de ligne à partir de zéro (par rapport au début de l'écran virtuel).
COLNUMBER	Numéro de colonne à partir de zéro
19	Code identifiant un champ attribut vidéo
ATTNUMBER	Code de l'attribut.
20	Code signalant le champ type de zone
BLOCTYPE	22 : Affichage protégé 23 : Affichage modifiable
CODE-STX	Indique le début du texte
texte	Suite de caractères à afficher
.	
.	S'il y a plusieurs zones à afficher
.	On recommence à CODE-SOH.
.	
CODE-EOT	

Remarque :
Un texte structure en ligne est une suite de zones d'une longueur maximale de quatre vingts caractères.

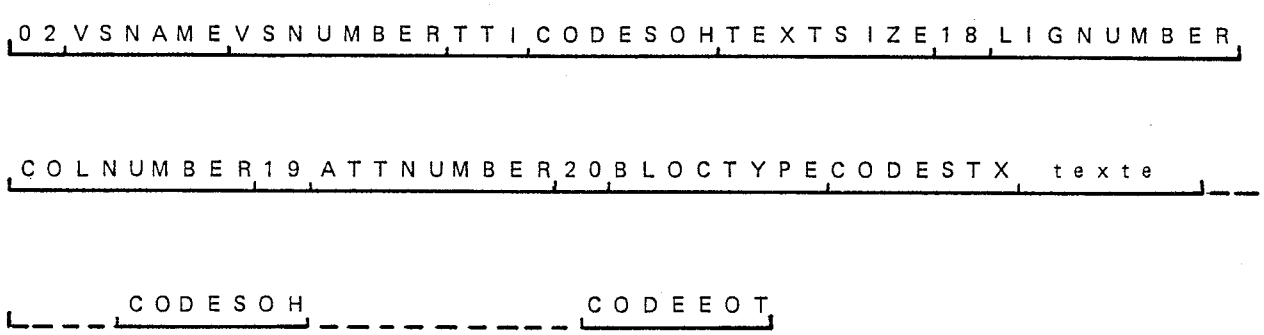


Fig. 5

La structure du message de compte-rendu est :

02	Indique un compte-rendu de 'DISP'
CRD	Compte-rendu d'erreur
VSNUMBER	Numéro d'écran virtuel
CODE-EOT	

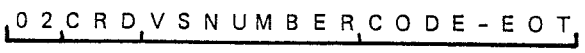


Fig. 6

Compte-rendu d'erreur :

0	: pas d'erreur
1	: erreur de syntaxe
3	: terminal hors service
4	: le numéro de l'écran virtuel et le nom sont incohérents (le VSNUMBER n'est pas celui retourné par la primitive CREATVS).
5	: Trop d'attributs vidéo pour une même ligne d'écran (10 au maximum) .

4.2.4 RECEIVE

Cette primitive permet d'acquérir une réponse de l'opérateur sous forme de :
Succession des choix faits pour les différents menus liés au même menu initial ainsi que les paramètres les étayants,
Ou, numéro d'une touche fonction (cf FUNCVAL),
Ou, commande destinée au module client.

Le message d'appel est structuré ainsi :

03	: Indique une primitive RECEIVE
VSNAME	
VSNUMBER	
CODE-EOT	

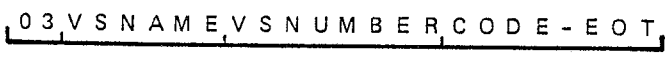


Fig. 7

Voici la description du message de retour :

03	Indique un compte-rendu de 'RECEIVE'.
CRD	Numéro d'erreur.
VSNUMBER	
FIRST-LINE-NUMBER	Numéro de la première ligne modifiée dans l'écran virtuel.
LAST-LINE-NUMBER	Numéro de la dernière ligne modifiée dans l'écran virtuel.
32	Signale que le message constitue une réponse à un menu.
NBCHOIX,	
CHOIX1,...,CHOIXn	idem
LGP1,	
P1,	compte-rendu
.	
.	De
.	
LGPn,	CREAT VS
Pn,	
Ou bien	
33	indique la frappe d'une touche fonction
FUNCTION-NUMBER	identification de la touche fonction frappée par l'opérateur
Ou bien	
34	signifie que l'opérateur achoisi de dialoguer à l'aide de commandes
Libellé de la commande	
35	Indique que la réponse opérateur se constitue d'un texte structuré (CF DISPLAY).
0,1	1 indique que c'est la dernière zone
LGBLOC	Longueur du bloc (de la zone).
Texte	texte contenu dans la zone.

Compte-rendu d'erreur :

0	: pas d'erreur
1	: erreur de syntaxe
3	: terminal hors service
4	: incohérence VSNAME/VSNUMBER
6	: Time-out

Remarques :

1) Les zones renvoyées au module client sont uniquement celles dites en affichage modifiables : le contenu des autres n 'ayant pu être modifié.

2) Le time-out, de deux minutes, évite qu'un opérateur garde trop longtemps et inutilement des ressources et dégrade les performances de SEDRIC.

3) Si le terminal est hors service, la primitive est ineffective et le message de retour est vide.

4.2.5 READVS

Permet au module client d'acquérir le texte mémorisé dans l'écran virtuel. Elle est intéressante lorsque SEDRIC indique en compte-rendu de RECEIVE que l'opérateur a appuyé sur une touche fonction. Elle permet au module client d'acquérir le texte qui a été saisi auparavant. Cela peut être mis à profit par un éditeur de texte.

Le message d'appel est structuré de la manière suivante :

04	Code de la primitive
VSNAME	
VSNUMBER	
FIRST-LINE-NUMBER	Numéro de la première ligne à lire (-1, si c'est la ligne courante)
LAST-LINE-NUMBER	Numéro de la dernière ligne à lire
CODE-EOT	

Fig. 8

Le message de retour est le suivant :

04	retour READVS
CRD	
VSNUMBER	
CODESOH	Indique le début de la description
.	
.	Le message est alors identique,
.	Dans sa structure, au message
.	D'appel de la primitive 'DISPLAY'.
.	
CODE-EOT	

Remarque :

Le message reflète la structure de l'écran. Il est donc composé d'un ensemble de lignes ou de zones, éléments d'une grille d'affichage.

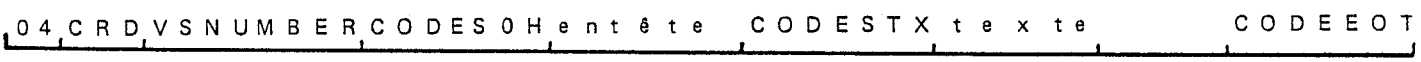


Fig. 9

Compte-rendu d'erreur :

0	: pas d'erreur
1	: erreur de syntaxe
4	: incohérence VSNAME/VSNUMBER

4.2.6 READ MODIFIED

Le rôle de cette primitive est voisin de la précédente. Elle est, cependant, d'une utilisation plus fine car elle fournit l'ensemble des lignes modifiées situées, dans l'écran virtuel, entre deux lignes dont on donne leurs numéros dans le message d'appel.

Ce dernier est le même que celui de READVS. Seul le code de la primitive change. C' est '05'.
Il en est de même pour le message de retour.

4.2.7 FUNCTION-VALIDATION (FUNCVAL)

Permet de valider certaines touches fonction et de leur donner une signification propre à chaque module client. Lorsque l'opérateur appuie sur l'une d'entre elles, on transmet, sur 'RECEIVE', son numéro. Les autres touches fonctions, non validées, restent sans effet.

Le message d'appel est le suivant :

06	Type de la primitive
VSNAME	
VSNUMBER	
FUNCNUMBER	Nombre de touches à valider
FUNCNUM1	Numéro de la première touche
.	
.	
.	
FUNCNUMn	Numéro de la dernière touche
CODE-EOT	

Remarque :
Aucune valeur ne pouvant être prise par défaut, il en résulte que tous les paramètres sont obligatoires

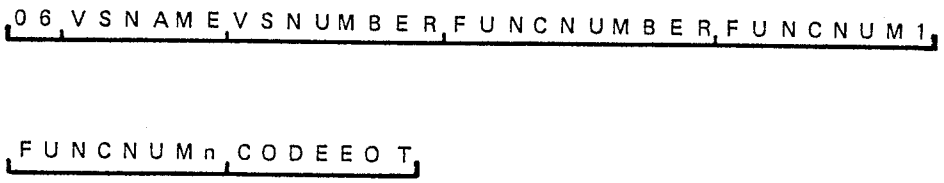


Fig. 10

Le message de retour est le suivant :

06	
CRD	Compte-rendu d'erreur
VSNUMBER	
CODEEOT	

Le compte-rendu d'erreur peut être :

00	: pas d'erreur
01	: erreur de syntaxe
04	: incohérence VSNAME|USNUMBER
07	: touche réservée à SEDRIC
08	: Nombre de touches disponibles insuffisant

4.2.8 DELETE VIRTUAL SCREEN (DELVS)

Conclut le dialogue, permet au serveur de désallouer l'écran virtuel et de libérer le segment d'écran.

Le message à fournir est le suivant :

10	: code de la primitive
05	: code VSNAME
VSNAME1	: nom du premier écran virtuel
VSNUMBER1	: numéro correspondant
.	On répète cela si plusieurs écrans
.	Virtuels figuraient dans la primitive
.	'CREAT'
VSNAMEn	: nom du dernier écran virtuel
VSNUMBERn	: et son numéro
CODE-EOT	

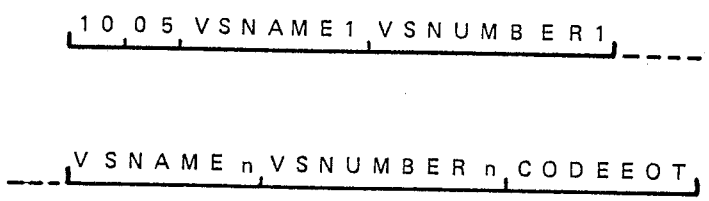


Fig. 11

Le message retourné est le suivant :

10
CRD
CODEEOT

Compte-rendu d'erreur

00	: pas d'erreur
01	: erreur de syntaxe
04	: il y a incohérence sur l'un des noms d'écran virtuel et le numéro associé.

4.3 L'INTERFACE USAGER

4.3.1 Généralités

Nous avons voulu au maximum alléger le travail de l'opérateur tout en le rendant maître de son dialogue. A aucun moment, il est contraint de subir les exigences du module client. Cela nuirait à l'opérabilité.
Pour l'assister il dispose de plusieurs touches fonctions prédéfinies que nous allons décrire maintenant.
L'activation de l'une d'elles provoque, la plus part du temps, la recopie du segment d'écran dans la fenêtre virtuelle. De ce fait, il y a une entière cohérence entre, les informations visibles par l'opérateur et celles données au module clients.
Ces fonctions sont au nombre de douze. Sur le terminal cible qu'est la DT15, elles sont matérialisées par les touches F1 à F11. Pour certaines les deux positions d'une même touche sont utilisées (shift, unshift).
L'opérateur dispose également des fonctions d'édition locales au terminal, et, dont SEDRIC n'a pas à se préoccuper.

4.3.2 MENU-HISTORIQUE

Lorsque le dialogue s'effectue à l'aide de MENU, un historique est assoc ié à chaque écran virtuel. C'est lui-même un menu. Quand l'opérateur appuie sur cette touche, SEDRIC l'affiche dans le segment concerné. Il contient l'intitulé de chaque item choisi au fur et à mesure des menus, avec les paramètres communiqués. Chaque item est précédé du numéro d'ordre du menu correspondant et de son nom.
L'opérateur peut ainsi, soit,

- demander un menu en fournissant le numéro de l'item l'identifiant, soit,
- modifier les paramètres de l'historique. Cela correspond à activer le dernier menu avec de nouveaux paramètres.

S'il n'y a pas d'historique, SEDRIC affiche le premier menu (racine de l'arbre des menus) associé à l'écran virtuel pour lequel on travaille.
Si le module client ne dialogue pas à l'aide de menu, la touche est sans effet.

Exemple :

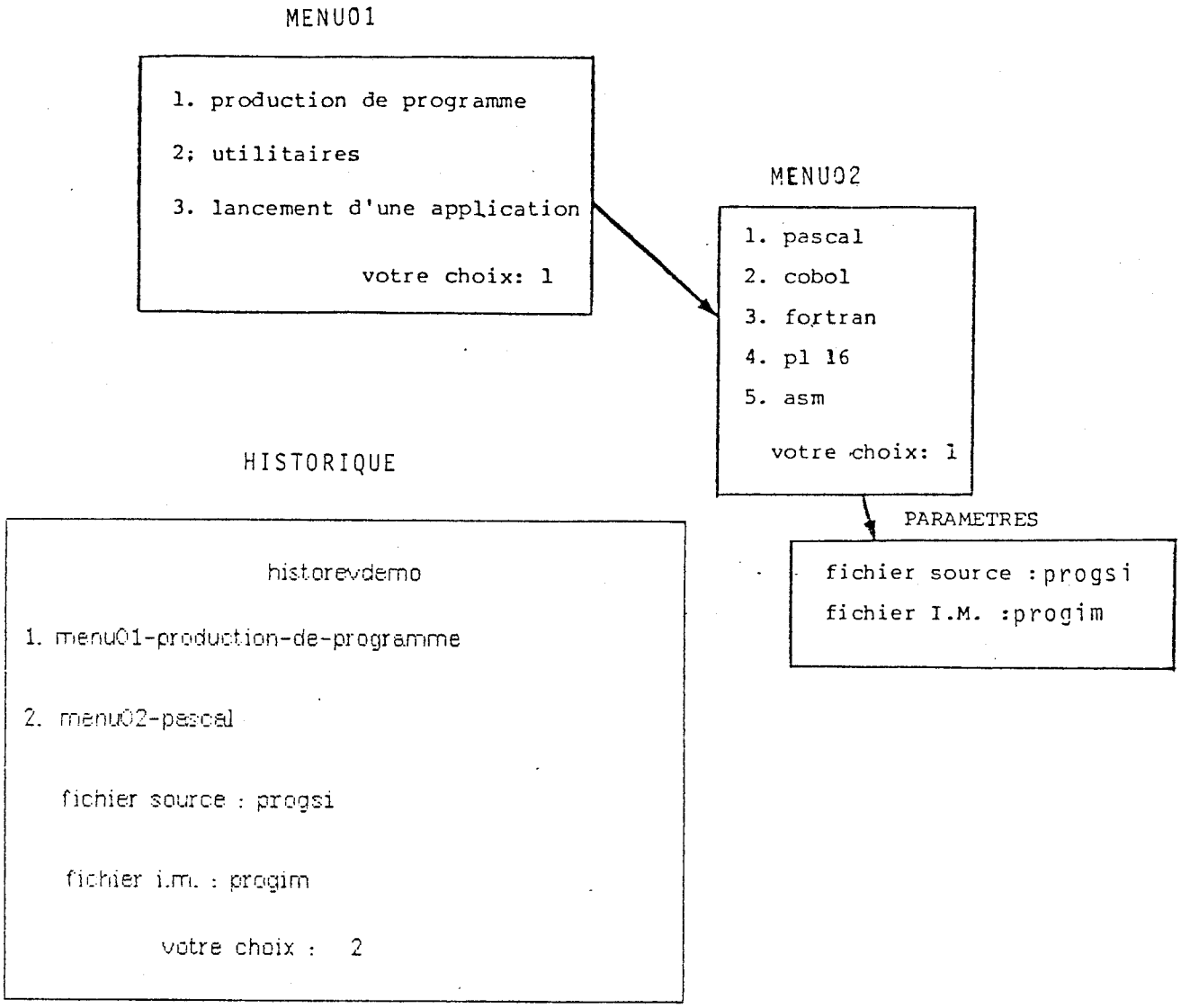


Fig. 12

En choisissant l'item numéro 2, l'opérateur sélectionne le deuxième menu. La liste des compilateurs est aussitot affichée.

4.3.3 COMMAND

Cette touche permet deux types de dialogues sans enfermer l'usager dans l'un ou dans l'autre. Ce ne sont pas deux environnements distincts. Elle est le complément de la touche "MENU-HISTORIQUE". Son but est d'interrompre un dialogue sous forme de menus pour permettre à l'opérateur expérimenté de fournir sa réponse à l'aide d'une commande.
Elle rend actifs un segment et un écran virtuel dédiés à cette tâche et dans lequel on mémorise la liste des commandes. L'opérateur dispose ainsi d'une véritable trace de son dialogue. S'il veut de nouveaux être guidé, il peut redemander un menu en pressant sur la touche 'MENU-HISTORIQUE'. Il faut néanmoins que la présence de ces deux modes de fonctionnement soient prévue et permise par le module client (CF CREATVS).
Si l'on interrompt un menu l'historique est cohérent : il relate la succession des choix jusqu'à l'interruption. Le message fourni au module client est constitué de la commande frappée.
Inversement, le message relatera les différents choix pris par la suite.

4.3.4 SEGMENT-SELECTION

Nous abordons maintenant la sélection de segments. Il y a deux manières de procéder.

Par menu : touche MENUSEL

Pour quitter l'activité en cours et, donc, une saisie ou un affichage intempestif, l'opérateur appuie tout d'abord sur la touche 'break', puis sur la touche MENSEL (Menu de sélection). Cela provoque l'affichage d'un menu composé de tous les noms des segments activables.

Ces noms sont formés :

- du nom de l'écran virtuel
- suivi du numéro du segment pour cet écran virtuel (CF Fonction 'DIVISION')

Ils apparaissent par ordre de création des segments.

Exemple :

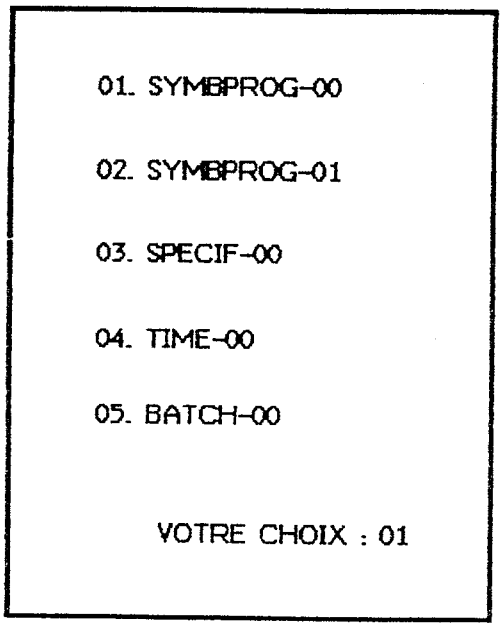


Fig. 13

En frappant le numéro 01, l'opérateur demande l'activation du premier segment.
Cette fenêtre physique est alors affichée avec son dernier contenu.

Remarque :
Le nom des segments est connu de l'opérateur car il apparait sur l'écran en entête de la fenêtre physique.

Par désignation : touche SELSEG

La sélection de segment par menu pésente l'avantage de permettre le choix d'un segment totalement caché. Elle a l'inconvénient d'être d'un usage relativement lourd. C'est pourquoi nous avons éprouvé le besoin de fournir un autre mode opératoire que nous allons décrire ici.
Pour les mêmes raisons que précédemment l'opérateur appuie tout d'abord sur la touche 'break'. Ensuite il se positionne sur le segment désiré à l'aide des touches de déplacement du curseur. Il appuie alors sur la touche SELSEG pour signaler son choix à SEDRIC.

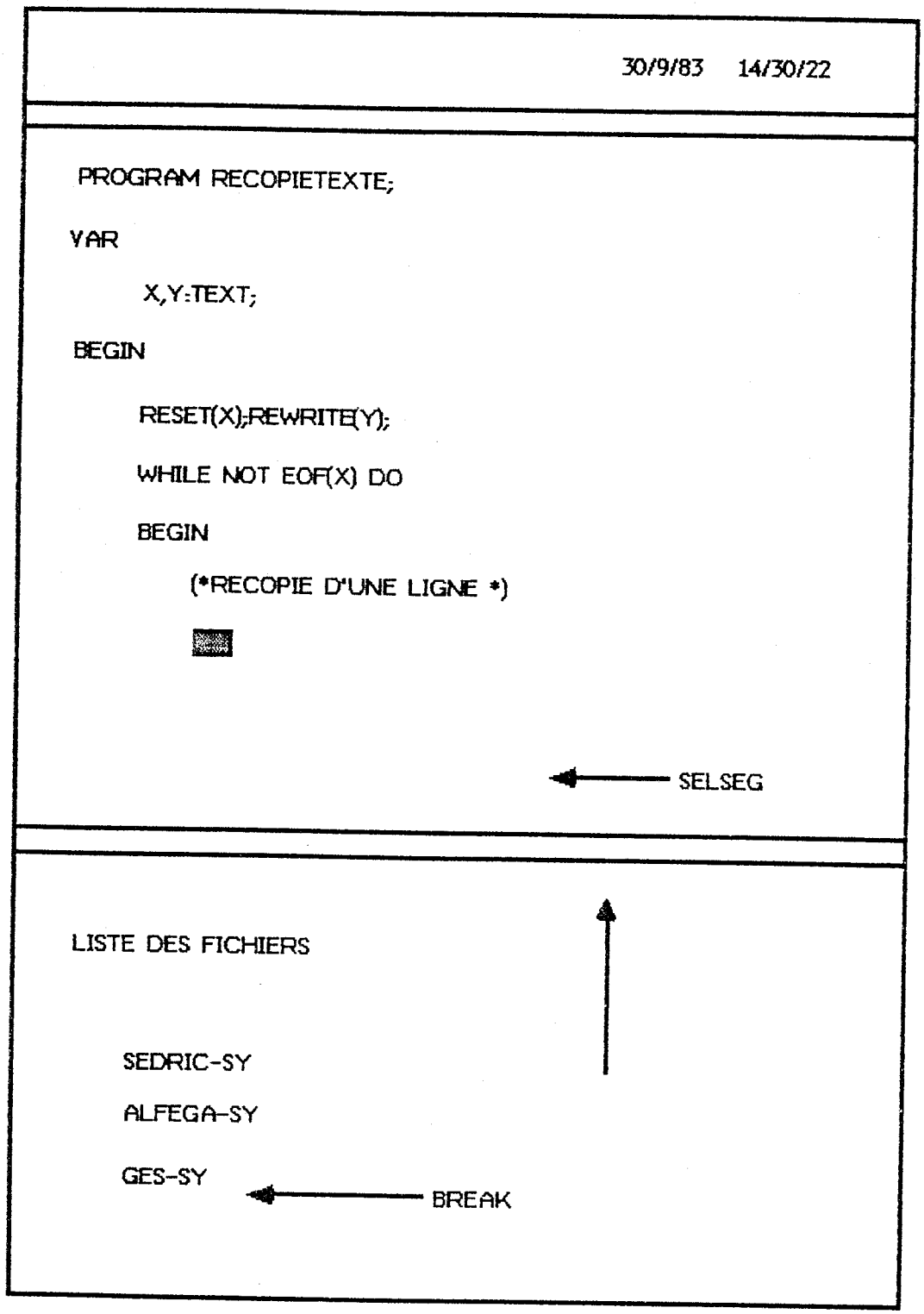


Fig. 14

4.3.5 SEND

Cette touche fonction permet de valider toute la saisie de l'opérateur depuis le dernier 'SEND' ou le dernier affichage. Il y a également mise à jour de l'historique et construction de la réponse si l'on travaille avec des menus.
Dans tous les cas, on affecte au module client une nouvelle fenêtre virtuelle.
C'est sur la frappe de cette touche que celui-ci reçoit la réponse à une primitive 'RECEIVE' ou 'CREATVS'.

4.3.6 WINDOW UP

Cette fonction est activée par l'opérateur lorsqu'il veut atteindre une ligne au-delà de la première ligne du segment. Il y a alors translation vers le haut de la fenêtre virtuelle d'un nombre de lignes appelé quantum qui dépend de sa taille (un tiers de la taille). Le nouveau contenu est ensuite affecté dans la fenêtre physique.

4.3.7 WINDOW DOWN

C'est la fonction récriproque. Elle est activée lorsqu'on veut afficher une ligne située en-dessous de la dernière ligne du segment. La fenêtre virtuelle est déplacée vers le bas du même quantum.

Exemple d'utilisation des fonctions 'WINDOW UP' et 'WINDOW DOWN' :

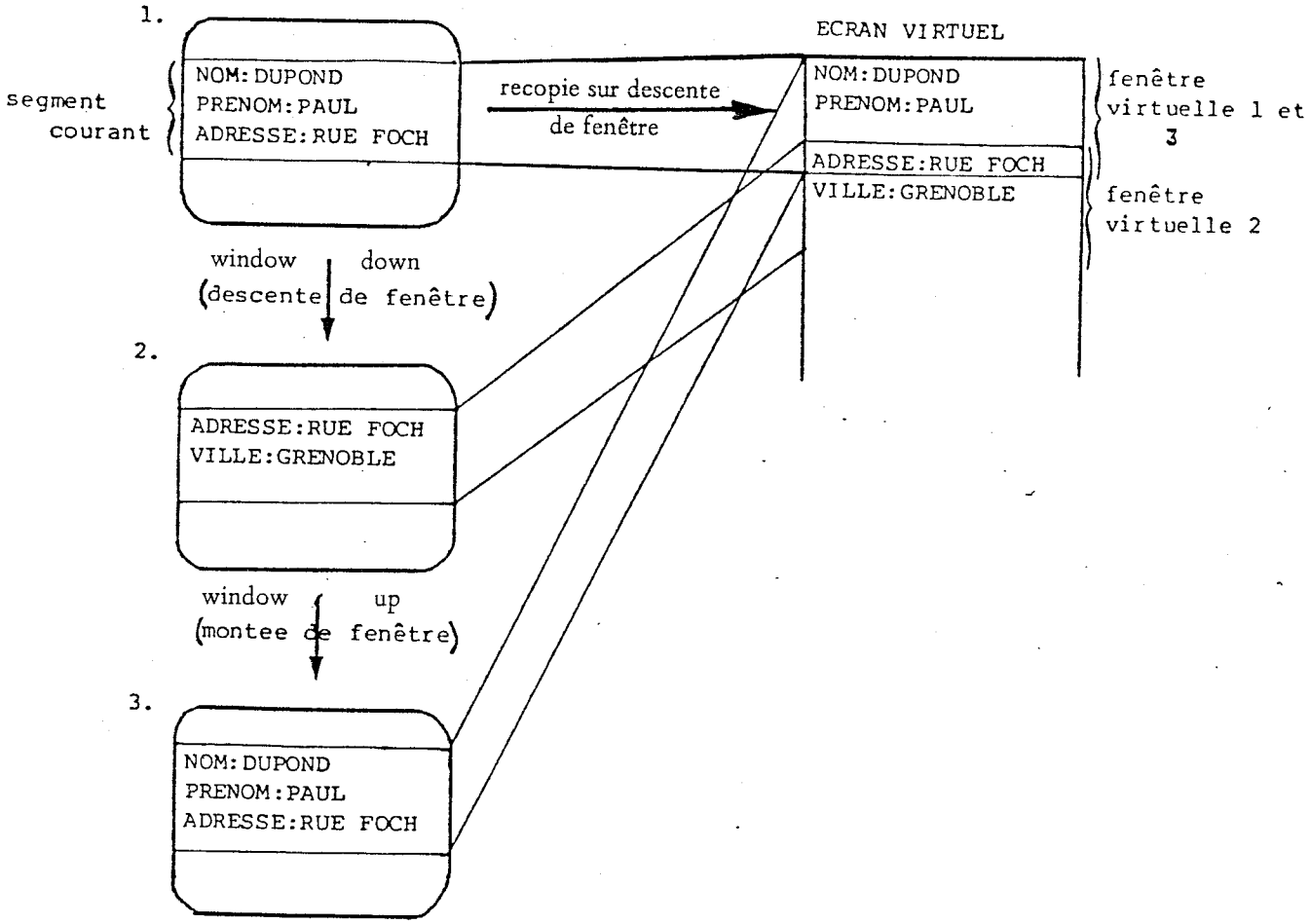


Fig. 15

4.3.8 ENLARGEMENT FROM TOP

Cette touche donne la possibilité à l'opérateur d'agrandir la taille du segment courant d'une ligne vers le haut. Il n'y a pas diminution des autres segments mais éventuellement recouvrement des uns, disparution apparente ou réapparition de certains. Parallèlement la fenêtre virtuelle associée est agrandie. Cette fonction est ineffective si l'on est au sommet de l'écran physique ou virtuel.

4.3.9 ENLARGEMENT FROM BOTTOM

La fonctionnalité est semblable à la précédente, mais l'agrandissement se fait vers le bas. Si l'on est au bas de l'écran virtuel, on en affiche les premières lignes. Si l'on est au bas de l'écran physique, la frappe de la touche est sans effet.

4.3.10 DIMINUTION FROM TOP

Permet de diminuer la taille du segment d'écran courant d'une ligne depuis le haut. Tous les segments touchés sont réaffichés.

4.3.11 DIMINUTION FROM BOTTOM

La différence avec la précédente fonction réside dans le fait que la diminution se fait depuis le bas.

4.3.12 DISPLACEMENT

A l'aide de cette touche, l'usager peut déplacer le segment courant à l'intérieur de l'écran physique.
Le déplacement se fait d'une ligne vers le haut ou vers le bas (shift).
SEDRIC effectue un nouvel agencement de l'écran pour afficher le segment à l'endroit désiré.

4.3.13 DIVISION

On éprouve souvent le besoin de visualiser en même temps, mais d'une façon ponctuelle, deux parties distinctes d'une entité d'un même module client. Il serait alors dommage de résoudre ce problème par la création de deux écrans virtuels pour la même activité et de gérer des informations souvent redondantes. On offre alors la possibilité de dédoubler un segment. Cela ne peut se faire qu'une seule fois par segment.
Lors d'une édition de texte pour modifier un programme, l'opérateur peut, ainsi, faire afficher les données dans le premier segment et les instructions dans le second. Il dispose des touches fonction SELSEG ou MENUSEL pour passer de l'un à l'autre.Dans le 'menu' des segments, ils ne sont alors différencier que par leurs numéros. Exemple :

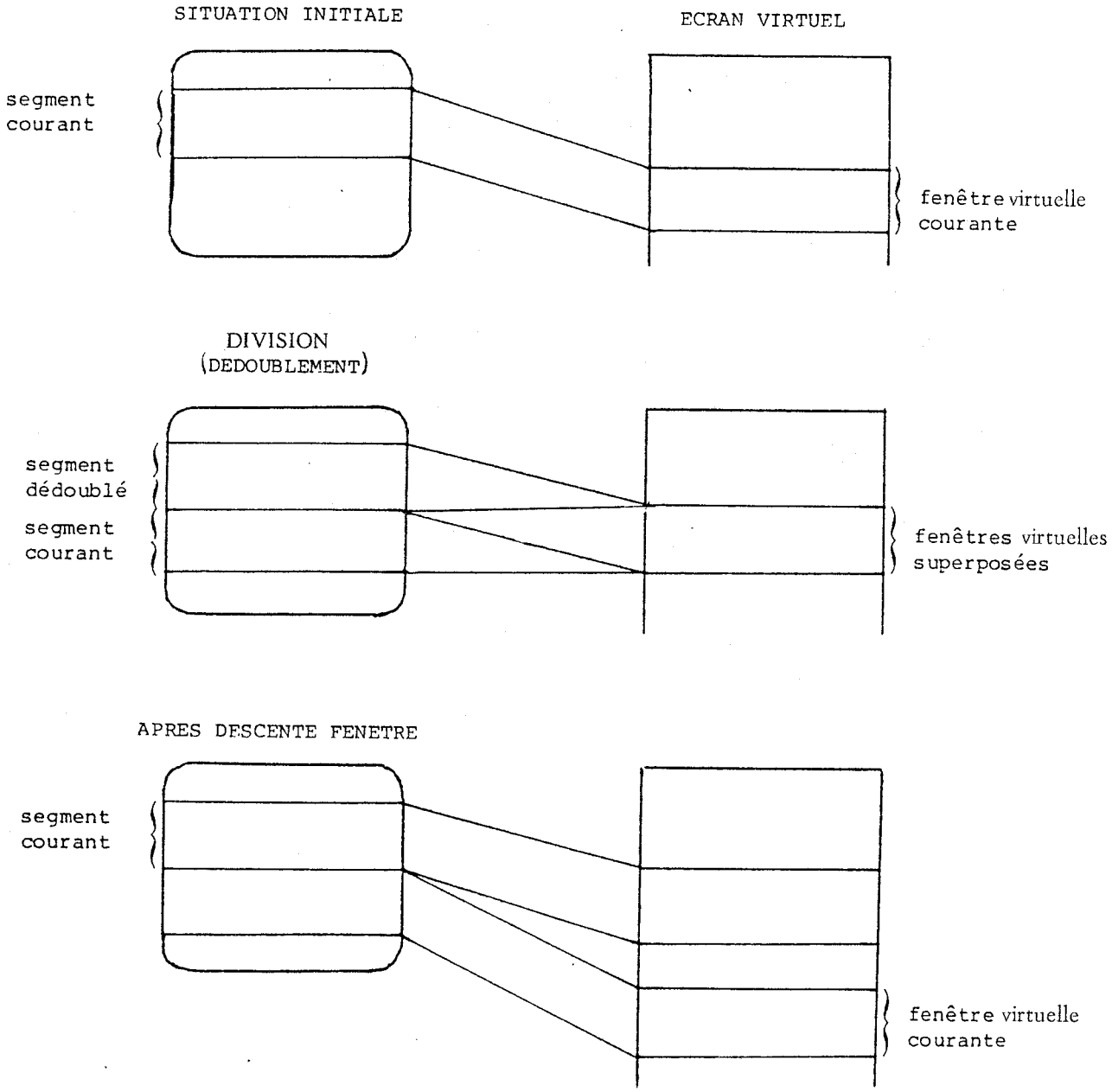


Fig. 16

5 REALISATION

5.1 GENERALITES

L'implémentation des concepts utilisés dans SEDRIC s'est faite à travers une maquette. C'est elle que nous allons décrire dans ce chapitre. Auparavant, nous présenterons le système support et nous justifierons son choix.
La totalité des primitives et de l'interface usager a été analysée et spécifiée. Cependant, pour des raisons de délai seules, à ce jour, les fonctionnalités les plus caractéristiques de SEDRIC ont été programmées testées et validées.
Ce sont:

- l'ensemble des primitives, excepté 'READ MODIFIED';
- le fonctionnement en multi-fenêtres, multi-activités et multi-consoles;
- la gestion des menus et des historiques.

5.2 LE SYSTEME SUPPORT

C'est le système temps réel de la gamme SOLAR. C'est-à-dire RTES/D (Real Time Executive Système ou Disk).

Nécessité d'un système multi-tâches

Dans un premier temps, il nous parut plus intéressant, dams le cadre d'une utilisation future, de réaliser SEDRIC sur un système temps partagé, en l'occurrence TSF ou TSM pour SOLAR.
Il y avait néanmoins deux inconvénients à cela.

1) La notion d'usager, au sens système, est liée au terminal. A une console correspond un utilisateur et un seul, donc un contexte d'exécution. Activités et fenêtres multiples devenaient alors difficilement réalisables. Il fallait opérer de profondes modifications dans le système.

2) A chaque utilisateur correspond une tâche et donc une priorité. De ce fait, leur nombre est limité (32 actuellement). Celui des activités simultanées s'en trouve réduit en conséquence.

On ne se heurte pas à ces problèmes avec un système multi-tâches.
Par contre, on perd certains avantages tels que la gestion des utilisateurs et de leurs contextes d'exécution.

Nous avons donc été conduit à concevoir notre maquette sur le système RTES/D que nous allons présenter maintenant.

5.2.1 Présentation du système RTES/D

Tâche utilisateur

L"unité de base du programme sous RTES-D est la "tâche"; elle consiste en un programme ou un ensemble de programmes écrits en langage FORTRAN, PL16, ASSEMBLEUR ou PASCAL.
Elle est produite sur disque par utilisation des processeurs du software de base (compilateur, éditeur de liens...); ultérieurement, elle est "chargée" sous RTES-D par une commande de l'opérateur qui l'introduit dans l'application.
Une tâche est alors identifiée en tant qu'entité fonctionnelle de RTES-D par sa priorité ou son nom (tâches mères - tâches filles) et son type (hardware ou software) et par son contexte qui regroupe les informations statiques et dynamiques .
Au niveau externe, une tâche est identifiée par son numéro d'appel; l'opérateur dispose de commandes d'intervention ON-LINE qui font référence à une tâche par son numéro d'appel; les autres tâches de l'application font référence à une tâche donnée, dans certaines requêtes programmées, également par son numéro d'appel. Il s'agit d'un numéro symbolique, la priorité étant au contraire un numéro fonctionnel.
Au moment de l'écriture de la tâche, l'usager lui assigne une "priorité et un type par défaut" : ce sont les caractéristiques fonctionnelles avec lesquelles elle s'exécutera si l'opérateur, par la commande appropriée, ne les modifie pas.
Le numéro d'appel, spécifié lors de la création de la tâche, est immuable.

Gestion mémoire

L'analyse des applications "temps réel" permet de dégager une structure partitionnée des différents travaux liés à l'application. Cette structure est schématisée ci-après.

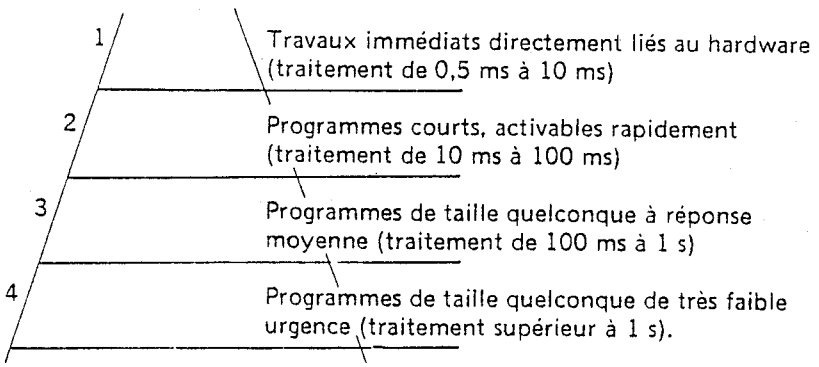


Fig. 17

La gestion proposée s'inspire de ce schéma. Les programmes de type 1 et 2 sont en général résidents en mémoire. Par contre, les programmes 3 et 4 sont résidents sur disque. D'où le schéma ci-après de la géographie mémoire.

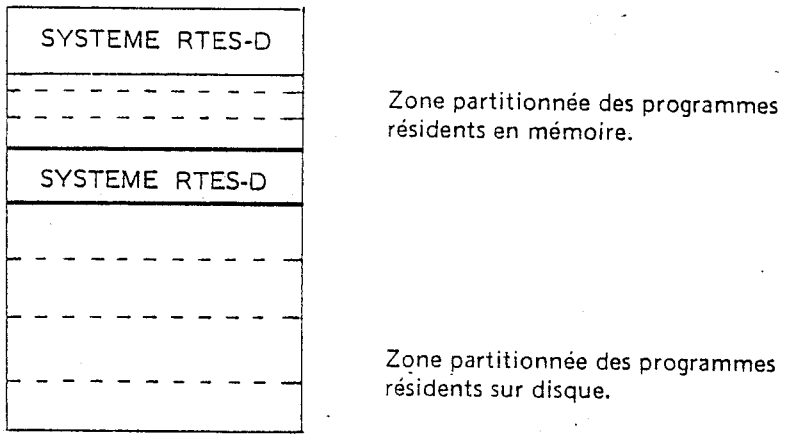


Fig. 18

La zone des programmes résidents-disque est découpée en partitions de tailles différentes. Ce partitionnement permet l'exécution des tâches de l'application suivant une hiérarchie utilisateur.
Une tâche qui occupe une partition est une tâche qui est censée se terminer dans un certain temps.
Les tâches s'exécutant dans une même partition sont en général exclusives.

Exécution dans une partition

On a vu qu'une partition est une zone contigue de mémoire centrale utilisée pour l'exécution de tâches.

Toutes les tâches utilisateur s'exécutent dans des partitions, qu'elles soient résidentes en mémoire ou résidentes sur disque, de type hardware ou de type software.

Au moment de l'écriture de la tâche, l'usager lui assigne une partition "par défaut" : c'est la partition dans laquelle elle s'exécutera normalement si l'opérateur, par la commande appropriée ne modifie pas ON-LINE cette affectation. Il peut alors allouer à une tâche une autre partition, ou une liste de partitions "possibles" (4 au maximum).

A l'exception des tâches nécessitant un temps de réponse minimum (tâches hardware en particulier, les tâches utilisateur sont d'une façon générale résidentes sur disque et sont chargées en mémoire centrale dans une partition avant leur exécution.

L'état de résidence d'une tâche est figé non pas à sa création, mais au moment de son introduction sous RTES-D.

Une tâche résidente sur disque libère sa partition en fin d'exécution; elle n'est pas réécr ite sur disque (pas de SWAP - OUT); lors de sa prochaine activation, si la partition antérieurement allouée à été occupée par une autre tâche, elle sera lue sur disque dans sa version initiale. D'une exécution à l'autre, une tâche résidente sur disque ne pourra donc pas se communiquer des données par des mémoires de travail.

Protection intertâche

Les tâches temps réel particulièrement critiques doivent s'exécuter dans un environnement présentant un maximum de sécurité vis à vis des autres tâches de l'application; de même, le moniteur, coeur de l'application, doit être intégralement protégé des tâches en cours de mise au point.
RTES-D utilise la protection intertâche cablée du SOLAR 16 (DRPS16). Le moniteur s'exécute en mode maître; d'une façon générale, les tâches de l'utilisateur (et les processeurs Background) opèrent en mode esclave et sont protégées individuellement. Pour exécuter les instructions privilégiées qui leur sont interdites, elles disposent de requêtes programmées au moniteur qui réalisent toutes les opérations de contrôle assurant à l'application un maximum de sécurité.
La protection intertâche est assurée par RTES-D sous forme de protection inter-partitions : au moment de l'activation d'une tâche, après lecture en mémoire depuis le disque, ses deux registres de protection sont initialisés avec les adresses absolues qui délimitent la partition occupée; l'espace de travail d'une tâche est donc celui de la partition qu'elle occupe. Le processus est le même pour les tâches qui résident en mémoire; cependant il n'est pas "dynamique", de telles tâches s'exécutent en permanence dans la même partition : les registres de protection sont donc initialisés au moment du chargement (unique) de la tâche en mémoire.

Etats d'une tâche

Ils caractérisent l'ensemble des tâches de type software.
Ils correspondent à l'état des files "système" du scheduler microprogrammé (cf Manuel de Présentation du SOLAR 16) et des files internes de RTES-D.

- Inexistante : la tâche n'a pas été intégrée sous RTES-D
- Armée : le système a enregistré au moins une demande d'activation immédiate
- En attente

ou masquée : la tâche est en attente d'une fin d'opération d'entrée-sortie, d'un accés à une ressource, d'un événement ou d'une combinaison d'événements, d'un délai...

- Eligible : armée et non masquée
- Non prête : la tâche est éligible, mais le système doit réaliser pour elle un "complément de scheduling" avant de la lancer (la charger en mémoire si elle n'est pas présente; attendre qu'elle soit validée si elle est inhibée)
- Activable : éligible et prête
- Active : la tâche activable de plus haute priorité est la tâche active.

- Les états de présence :

- Résidente : la tâche réside en mémoire centrale; cet état est figé lors de l'intégration de la tâche sous RTES-D
- Présente : une tâche résidente est par définition toujours présente. Une tâche non résidente libère sa partition par la requête CALL EXIT; elle devient non présente dès que cette partition est allouée à une autra tâche.

Structure d'"OVERLAY"

Les tâches de l'application de niveau software peuvent être structurées en "OVERLAY". Une telle tâche se compose :
. D'une racine
. De N branches (0,255)

Une tâche en "overlay" est exécutée dans une partition dont la taille minimale est égale à la taille de la racine plus la taille de la plus grande branche (taille de l'Image Mémoire)

Le chargement en mémoire d'une branche se fait par écrasement de la précédente.

La racine est présente en mémoire pendant toute la durée du déroulement de la tâche; l'appel d'une branche provoque l'écrasement de la précédente sans sauvegarde sur disque; le transfert des paramètres entre branches doit se faire par les registres, par le système de fichiers ou par les données communes de la racine.

L'activation et l'abandon d'une tâche en "overlay" se font dans la racine; la tâche est reprise à l'instruction qui suit la demande d'abandon précédente.

L'activation par le scheduler comporte au plus 2 phases :

- chargement de la racine (cas d'une tâche non présente)

- lancement de la tâche au point de reprise dans la racine

Le lancement d'une branche par la racine ou une autre branche comporte au plus 2 phases :

- chargement de la branche

- lancement à une adresse fixe (cas de la première activation d'une branche).

Ressources de l'usager :

La ressource représente le moyen utilisé par la tâche; elle permet la coexistence et l'exécution parallèle des tâches.
On distingue :
- les ressources "hardware" qui appartiennent au système et dont il vérifie l'utilisation afin d'éviter tout blocage du système.
Exemple : les périphériques (ressources discrètes) la mémoire (ressource continue)
- les ressources "software" c'est l'ensemble :
  . Des services de base du moniteur accessibles par requêtes programmées (événements, horloges software...)
  . Des ressources créées par l'usager, qui représentent ce qu'il désire, dont l'utilisation est laissée à son entière responsabilité, et dont il se sert pour résoudre ses problèmes d'exclusion entre tâches.

Dans RTES-D, l'usager dispose de trois requêtes programmées pour créer des ressources software (en donnant leur nom et le nombre d'accès simultanés autorisés), demander un accès à ces ressources, et libérer un accès à ces ressources.

Communications entre tâches :

Le fonctionnement asynchrone des tâches d'une part, le fait qu'elles résident en mémoire centrale ou sur disque d'autre part, le fait enfin que la notion de protection intertâche leur impose des espaces de travail en mémoire disjoints, rendent indispensable l'existence de moyens souples et puissants d'intercommunication.
RTES-D présente divers outils complémentaire pour répondre à cet objectif.

La zone des données résidentes :

La zone commune des données résidentes constitue un moyen complémentaire du Système de Fichiers pour la communication entre tâches.
L'organisation de la zone des données résidentes est à la charge de l'usager; elle résulte de conventions entre les tâches de son application.
Aucune protection n'est réalisée par le système; l'usager doit synchroniser les tâches faisant accès à cette zone pour gérer les informations qu'elle contient.
Pour l'accès à ces données résidentes, les tâches disposent de 2 requêtes :
- REDGET, pour lire dans un buffer une portion de la zone des données résidentes.

- REDPUT, pour transférer un buffer dans une portion de la zone des données résidentes.

Paramètre de travail et compte rendu de traitement :

Au moment de l'appel d'une tâche, la tâche appelante ou l'opérateur peuvent préciser la nature du travail à réaliser à l'aide d'un paramètre de travail (un mot de 16 bits).

Ce paramètre peut ne pas être précisé : cette absence lui confère alors la valeur nulle.

Réciproquement, la tâche appelée peut renvoyer un compte rendu de traitement à l'appelant, sous la même forme d'un mot de 16 bits.

La zone de communication : CDA

La zone de communication est standard sur SOLAR 16-65. Elle permet une communication de données inter-tâches aisée quel que soit le mode de fonctionnement (mode maître ou mode esclave).
Comme pour la ZDR, les tâches utilisatrices de la CDA assurent elles-mêmes leur synchronisation.
Pour l'accès à ces données communes les tâches disposent :
- des instructions RCDA et WCDA (read CDA et write CDA)
- des entrées-sorties directes en CDA par le moniteur IOCS
- des entrées-sorties directes en CDA par le moniteur FMS
- de l'intégration et du lancement de tâches dans la CDA.

L'emplcement mémoire de la CDA est défini à la configuration du système par l'utilisateur.
Elle est constituée d'une ou plusieurs partitions contigues, ne peut pas dépasser 64 K mots et doit appartenir à la même page de 64 K.

Echanges de blocs d'informations

Cet outil de communication inter-tâche est basé sur l'utilisation de l'allocateur mémoire de RTES-D.
Pour réaliser ces échanges les tâches disposent de 2 requêtes programmées :
SEND pour envoyer un bloc d'informations dans la zone système (36 mots)
RECEIVE pour recevoir un bloc d'informations stocké dans le système (36 mots).
La gestion des blocs transmis par SEND et RECEIVE est réalisée par le système et les informations d'un bloc ne sont pas partageables .

Evénements :

Un événement est une donnée qui correspond à "quelque chose que l'on peut attendre", "quelque chose qui peut arriver". Il est sans mémoire et ne garde pas trace de ses différentes apparitions.

Plusieurs tâches peuvent attendre l'arrivée d'un même événement : par définition, un événement est partageable.

Un événement a deux états :

- l'état SET; un événement est à l'état SET quand il est arrivé

- l'état RESET; un événement est à l'état RESET quand il a disparu.

RTES-D gère 256 événements (numéros 0 à 255); Ces évènements sont scindés en 8 classes disjointes de 32 évènements chacune :

L'interprétation que fait l'usager des 32 événements d'une classe est liée à son application, et dépend des opérations qu'il désire réaliser sur une classe d'événements.

Pour utiliser les événements, on dispose de 8 requêtes programmées.

Phase d'avancement

La phase d'avancement est un outil d'intercommunication entre tâches utilisateur; elle permet de savoir dans quelle phase logique de son déroulement se trouve une tâche.

La phase d'avancement consiste en un numéro entre 0 et 255; par une requête programmée, une tâche précise au système, à chaque franchissement de points stratégiques, quelle est sa phase d'avancement.

Par une seconde requête programmée, une autre tâche peut savoir quelle est la phase d'avancement d'une tâche dont le numéro est spécifié en paramètre.

La phase d'avancement a une signification spécifique de chaque tâche qui résulte d'une convention entre deux ou plusieurs tâches utilisateur.

Cette notion est par exemple relativement intéressante pour un redémarrage après disparition du secteur : après avoir effectué un certain nombre d'actions standard, RTES-D active une tâche spécialisée qui peut prendre des décisions en fonction de la phase d'avancement de chaque tâche du système.

Les requêtes permettant l'accès à la phase d'avancement sont :

- STADEF : la tâche appelante définit sa phase d'avancement

- STAGET : La tâche demande quelle est la phase d'avancement d'une tâche précisée en paramètre.

Background de RTES-D

Caractéristiques générales

La fonction background de RTES-D est réalisée par un moniteur présentant les services et la souplesse du système d'exploitation BOS-D (système d'exploitation de base du SOLAR) et d'une utilisation compatible.

Cette fonction permet en particulier :

- d'utiliser simplement tous les processeurs du software de base .

- d'exploiter les programmes produits indifféremment par un des systèmes de la série BOS ou en Background,

- d'utiliser le système d'entrées-sorties IOCS et le système de fichiers FMS intégrés à RTES-D.

- d'exploiter des programmes en mode train de travaux ou conversationnel, ces programmes pouvant être structurés en overlay, c'est à dire ayant une taille supérieure à la taille mémoire occupée.

- d'utiliser les bibliothèques système ou utilisateur.

- de produire sur disque des tâches ultérieurement intégrées sous RTES-D, c'est à dire de faire évoluer une application par adjonction ou modification de tâches en background.

Cette fonction est optionnelle.

Dialogue système-opérateur

L'opérateur exerce un ultime contrôle sur l'application par l'intermédiaire de commandes d'une part et des messages et diagnostics d'erreurs qui lui sont signalés d'autre part. Cette fonction de dialogue, puissante et intéractive, est vitale pour l'application, et donc assortie d'un maximum de sécurité (mot de passe, messages d'erreurs...).
L'opérateur peut à l'aide de ces commandes :

- introduire en ligne de nouvelles tâches
- modifier l'heure
- initialiser le système
- visualiser des états du système, les sauver sur disque.

5.2.2 Gestion des entrées-sorties

Toutes les opérations d'entrée/sortie sont réalisées par le moniteur IOCS (Input Output Controle Systeme)
Cette interface, compatible pour l'ensemble des systèmes d'exploitation, est en outre modulaire : l'utilisateur peut aisément lui ajouter de nouveaux drivers de périphériques.
Ainsi RTES/D présente des drivers et des requêtes adaptés au fonctions industrielles.

Caractéristiques d'IOCS

Tout utilisateur s'adressant au moniteur bénéfice de :
- la standardisation des entrées/sorties,
- de l'indépendance des échanges vis à vis des périphériques et de plusieurs modes de gestion (canal, interruption, programmé prioritaire).
- de la réentrance d'IOCS (un seul module d'échange permet la gestion de plusieurs périphériques de même type.
- de l'existence de modules d'échanges (driver) relatifs aux périphériques portés au catalogue SOLAR.

IOCS est composé
- d'un noyau
- d'un certain nombre de modules d'échange relatifs aux différents type de périphériques gérés

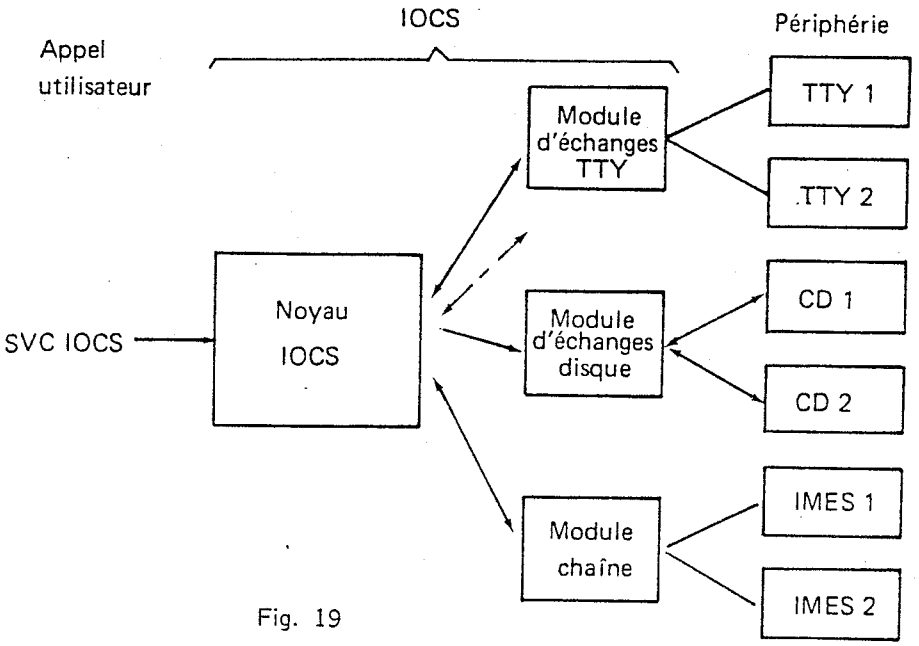


Fig. 19

Gestion du disque

Une caractéristique de RTES/D est l'unicité d'accés au disque quelles que soient les fonctions à réaliser :

- traitement des fichiers de l'usager
- alimentation des tâches en mémoire
- overlay sur les branches d'une tâche

Cette unicité du type d'accés, réalisée par l'intermédiaire du moniteur FMS (File Management System), assure :

- l'optimisation d'utilisation de l'espace disque
- la protection d'accés au divers fichiers
- la modularité du moniteur basée sur un environnement disque.

Caractéristiques de FMS

- L'allocation dynamique de l'espace disque assure l'indépendance des programmes vis à vis du volume des données à traiter et permet la récupération immédiate de l'emplacement occupé par les fichiers supprimés.

- FMS effectue une gestion banalisée de toutes les unités disque et simultanée des disques à cartouche et à tête fixe.

- FMS fournit en standard 3 méthodes d'accès qui sont :

  - séquentiel,
  - direct : articles de longueur fixe accessible par numéro,
  - indexé : articles de longueur variable accessible par nom

L'accès séquentiel est possible à l'intérieur des articles des fichiers directs et indexés.

FMS fournit en option :

- le séquentiel indexé :
  Qui gére un grand nombre d'articles de taille constante identifiables par une chaine de caractères.
- Le séquentiel chainé :
  Qui permet de constituer des chaines d'articles de largeur fixe.
- La bufferisation des entrées/sorties séquentielles :
  Qui diminue le nombre d'accés disque
- L'accés direct rapide :
  Qui donne accès à l'information en un seul accés disque

Les fichiers sous FMS

Ils peuvent être :

- temporaires (intéressant en production de programmes)
- permanents et partageables simultanément ou non.
  Le partage est controlé et adapté aux contraintes des systèmes temps réel ou multiconsoles.

Les fichiers indexés

Nous abordons de façon plus détaillé ce type de fichier car il est utilisé par SEDRIC pour stockés les menus.

On distingue 2 parties dans un fichier indexé.
- La table d'index : contenant des informations système gérées par FMS.
- Les articles du Fichier : contenant l'information de l'utilisateur.

A) Les articles : Un fichier indexé est composé d'un ou plusieurs articles identifiés par un nom. Le nom et la taille d'un article sont définis par l'utilisateur à la création de l'article. Un nom d'article est composé de 8 caractères ASCII (2 caractères par mot). FMS accepte une plage de 40 caractères : les lettres (A; Z), les chiffres (0; 9), ":", ";", "<", Nul (en fin seulement). FMS vérifie la non homonymie des noms d'articles. La

taille des articles est variable . Elle doit cependant être un nombre pair d'octets, inférieure à 64 K mots.

B) La table d'index : elle contient les noms des articles ainsi que l'adresse de chacun dans le fichier. C'est à la création du fichier que l'utilisateur définit le nombre maximum d'articles que le fichier pourra contenir. La taille de la table d'index est figée à la création du fichier par FMS :

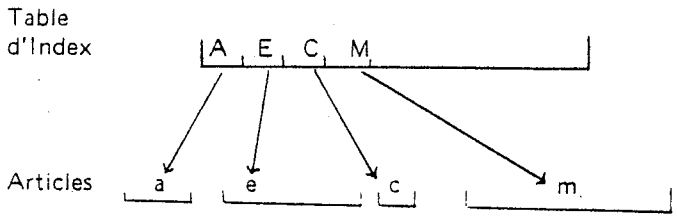


Fig. 20

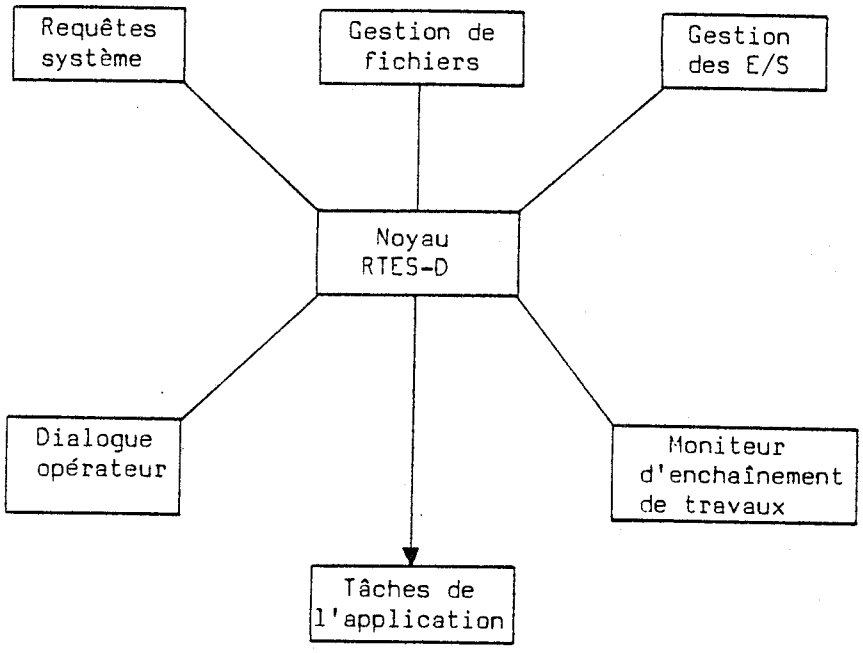


Fig. 20 bis Organisation générale de RTES-D

5.2.3 Les consoles de visualisation

Un seul type de consoles a été mis à notre disposition pour réaliser la maquette. C'est la "Télévidéo" trés proche de la VT100 de Digital Equipment, elle se place dans la catégorie des "smarts terminals".

Le terminal, encore appelé DT15, est un ensemble écran clavier connectable à un calculateur par une liaison série asynchrone respectant la norme RS232 (V24 ou boucle de courant) de 50 à 19200 bands.
Il peut fonctionner en mode :

Local
Caractéres (half ou full duplex)
Bloc (ligne ou page)

Il dispose également d'une sortie série asynchrone (V24 uniquement) pour le raccordement d'une imprimante.

Il permet la visualisation des 96 caractères du code ASCII, de 32 caractères de contrôle et de 15 caractères graphiques. Il autorise la visualisation de 24 lignes de 80 caractères. Une vingt cinquième ligne donne l'état du terminal, elle est modifiable par l'opérateur.

Deux groupes de dix commutateurs, en face arrière du terminal, permettent de cabler les vitesses des deux lignes et le mode de fonctionnement du terminal.

Programmation du terminal

C'est là l'un de ces avantages. Il peut traiter environ soixante commandes différentes qui vont du "bell" à la structuration des données sur l'écran. Cette programmation se fait à l'aide de séquences ESCAPE ou de caractères de contrôle émis depuis le calculateur ou le clavier.

. Caractère de contrôle : c'est un caractère dont le code est compris entre 00 (NUL) et 31 (US). Il est obtenu a partir du clavier en pressant simultanément la touche CTRL et une touche alphanumérique. Le code obtenu correspond aux 5 bits de poids faible du caractère alphanumérique.

. Séquence ESCAPE :
C'est une séquence de N + 2 caractères codée comme suit :

- Le premier octet est toujours le caractère de contrôle ESC (ESCAPE, code 27)
- Le deuxième octet identifie la fonction à réaliser.

Les touches fonctions programmables

Les touches F1 à F11, utilisables avec ou sans SHIFT, donnent accés à 22 fonctions supplémentaires programmables par l'utilisateur ou le calculateur. Une mémoire de 256 octets est réservée pour la programmation de l'ensemble des touches. A ces touches peuvent être affectés des chaines de caractères, des séquences ESCAPE ou des caractères de contrôle fréquemment utilisés. Lors de l'assignation d'un ou plusieurs caractères à une touche, il faut préciser le destinataire (écran, calculateur ou les deux).

On peut admettre que c'est également une manière de programmer ce terminal. Cette fonctionnalité est largement utilisé par SEDRIC.

Les différentes modes de transmission

Ils sont au nombre de quatre :

- local
- bloc
- half duplex
- full duplex

En mode local, tout caractère frappé au clavier est affiché sur l'écran mais aucun code n'est transmis au calculateur, et tout code transmis par le calculateur est ignoré par le terminal.

En mode bloc, tout caractère frappé au clavier est affiché sur l'écran sans transmission de code au calculateur. Lorsque le texte est composé, l'opérateur demande sa transmission (ligne ou page) en un seul bloc. Toutes les informations transmises par le calculateur sont affichées sur l'écran sauf pendant les transmissions "terminal vers calculateur" ou elles sont ignorées. C'est le mode utilisé par SEDRIC.

En half duplex, tout caractère frappé au clavier est à la fois transmis au calculateur et affiché sur l'écran.

En full duplex, les codes des caractères frappés au clavier sont uniquement transmis au calculateur. C'est à celui-ci de décider s'il doit renvoyer ou non l'écho du caractère.

A la mise sous tension, le mode validé dépend de la position des commutateurs.

Les modes d'affichage

- Le mode demi-intensité :
Tous les caractères reçus, depuis le clavier ou le calculateur sont affichées avec une luminosité moindre.

- Le mode protégé :
Lorsque ce mode est validé, tous les caractères affichés en demi-intensité, et certains autres tels que ceux définissant un attribut vidéo sont protégés. Seules les parties de l'écran non protégées peuvent recevoir des caractères depuis le clavier ou le calculateur. Seules ces zones peuvent être transmises vers l'ordinateur (touche SEND de l'interface usager)

Les attributs vidéo

Le terminal permet la visualisation en vidéo inversée, le soulignement, le clignotement et la non visualisation. Tous ces attributs peuvent être combinés. La commande définissant l'attribut visuel est codée sur trois octets et occupe un octet dans la mémoire de l'écran. Cet octet attribut est visualisé par un espace noir lorsque le fond de l'écran est noir, ou par un espace en demi-intensité lorsque le fond de l'écran est vert. L'attribut visuel porte jusqu'à l'attribut visuel suivant ou, à défaut, jusqu'à la fin de l'écran. Lorsque le mode protégé est validé, les octets contenant les attributs visuels sont protégés. Le tableau ci-dessous donne les commandes reconnues par le terminal.

ESC G 0 Normal

ESC G 1 Invisible

ESC G 2 Clinotant

ESC G 3 Invisible - Clignotant

ESC G 4 Inversé

ESC G 5 Inversé - Invisible

ESC G 6 Inversé - Clignotant

ESC G 7 Inversé - Invisible - Clignotant

ESC G 8 Sousligné

ESC G 9 Sousligné - Invisible

ESC G : Sousligné - Clignotant

ESC G ; Sousligné - Invisible - Clignotant

ESC G < Inversé - Sousligné

ESC G = Inversé - Sousligné - Invisible

ESC G > Inversé - Sousligné - Clignotant

ESC G ? Inversé - Sousligné - Invisible - Clignotant.

Nous allons maintenant décrire l'implémentation de SEDRIC sur SOLAR. Nous commencerons par les menus.

5.3 LES MENUS

5.3.1 Généralité

Nous invitons le lecteur désireux de se remémorer la définition d'un menu à se reporter au chapitre terminologie.

Un menu est référencé grâce à son nom. C'est une suite d'au plus huit caractères auquels peuvent s'ajouter (:, ;, <).
C'est un article d'un fichier indexé au sens FMS.
Ce fichier contenant tous les menus et toutes les grilles de paramètres s'appelle MENU-SD.

5.3.2 Structure d'un menu

Un menu quelconque est caractérisé par trois types d'éléments :

- les items du menus :
C'est leur liste qui présente des choix à l'opérateur. Chaque item est constitué d'un groupe de mots ou de symboles, c'est à dire d'une suite de caractères, de longueur limitée; ils constituent le libellé. Les items peuvent être en nombre variable illimité. Un menu peut-être vide. C'est le cas parfois de l'historique.

- les menus fils :
Lorsque l'opérateur sélectionne un des items, ce choix, s'il n'est pas terminal, produit l'affichage d'un autre menu, le menu fils, ou d'une grille de paramètres.

- le titre du menu :
Quand on présente à l'opérateur la liste des items, il peut lui être agréable de disposer d'un titre qui l'informe sur la nature du menu. Ce titre, qui, comme le libellé des items, est fourni par l'usager à la création du menu, est généralement différent du nom du menu lui-même. On a interêt à ce qu'il soit plus explicite. En effet, lorsque par la suite le menu sera utilisé, seuls les items et le titre apparaîtrons à l'opérateur. Le nom du menu sert uniquement à SEDRIC, pour connaître le menu à afficher.

Présentation

En plus des entités propres au menu lui-même, il existe d'autres particularités à retenir et qui sont plutôt d'ordre esthétiques, à savoir :

- la manière dont sont disposés les libellés des items et le titre dans la fenêtre (numéro de ligne, de colonne, disposition relative des différents groupes de caractères, attribut vidéo).

Toutes ces données ne doivent pas être figées une fois pour toutes. Au contraire, l'usager doit pouvoir choisir aisément comment pésenter son menu lorsqu'il le construit.

Rappelons enfin qu'un menu à une structure arborescente qui peut devenir un graphe sans cycle dans certain cas :

Le choix d'un item conduit à afficher un menu appartenant à une autre branche.

Toutes ces considérations ont conduit à adopter une structure très souple dont en voici le détail.

Description d'un article du fichier MENU-SD

Cette description est une description élément par élément

.0	: menu ou grille (code menu=0, code grille=1)
.1	: nombre d'items
.2	: adresse de la table d'adresses des items
.3	: adresse de la table de décision sur choix
.4	: adresse de description du titre
.5	: adresse du descripteur de la variable réponse
.6	: adresse de fin d'article
.7	: indicateur de saturation des tables (=nombre d'items max).
.8	: adresse du premier trou
.9	: libre

Ce sont ces dix premiers mots qui sont figés; à partir de maintenant, la description se fait selon les adresses indiquées précédemment.

.10	: longueur du titre
.11	: attribut vidéo
.12	: position du texte (ligne) ..... Description
.13	: position du texte (colonne) .... Du titre
.14	: adresse du texte
.15	: texte

Puis se suivent en général successivement :

Description de la variable réponse (qui permet par un numéro d'indiquer quel choix est effectué lors du déroulement du menu),
Table d'adresses des items,
Table de décision sur choix,
Description de chaque item l'un après l'autre.

Descripteur de la variable réponse :

- Longueur de la variable
- Attribut vidéo
- Position de la variable (deux mots)
- Adresse du texte
- Texte

Table d'adresses des items :

Cette table contient autant d'entrées qu'il y a d'items; l'entrée i donne l'adresse de la description de l'item i.

Table de décisions sur choix :

Cette table contient un élément par item; cet élément est le nom du menu qui correspond à l'item considéré; c'est donc aussi le nom d'un article du fichier MENUS-SD. Ce peut être égale ment le nom d'une grille de paramètre.

Descripteur de l'item i :

- Longueur de l'item i
- Attribut vidéo
- Position du texte (deux mots)
- Adresse du texte
- Texte

La structure des grilles de paramètres a été étudiées pour être compatible avec celle des menus. Elle présente néanmoins quelques différences.

L'élément 3 est sans signification.
L'élément 5 contient l'adresse de la table d' adresses des descripteurs des variables paramètres. Ce sont les zones d'écrans dans lesquels ont saisira les paramètres.
Les éléments 6 et 8 sont sans signification.
L'élément 9 indique le nombre de paramètre effectifs.

Exemple de menu et liste des informations dans l'article .

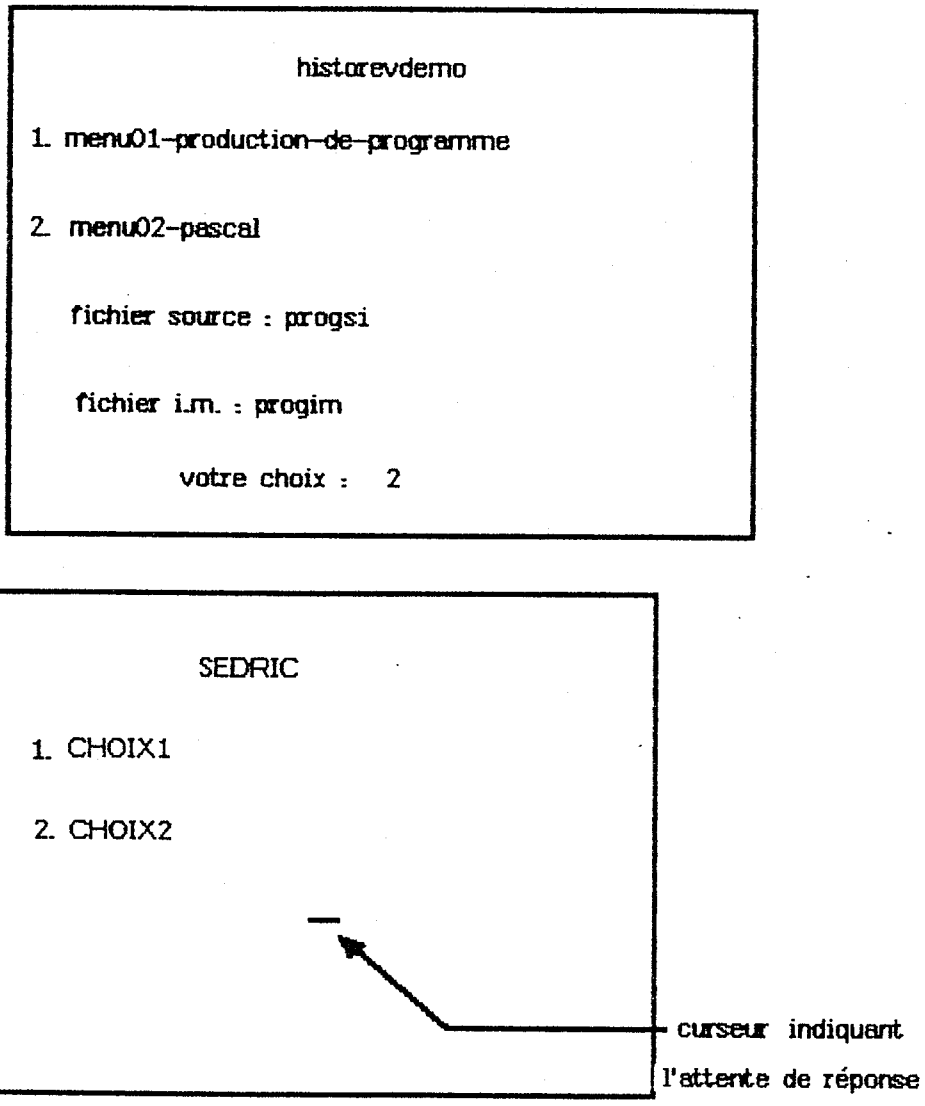


Fig. 21

Remarque :

Les adresses sont relatives au début de l'article, elles sont en fait des index.

```
0000                    : menu
0002                    : 2 items
0016                    : adresse de la table d'adresse des items (index
                          dans le menu)
0018                    : adresse de la table de décision
000A                    : adresse du descripteur de titre
0012                    : adresse du descripteur de la variable réponse
0030                    : adresse fin d'article
0002                    : indicateur de saturation : le nombre d'item
                          maximum est atteint
0000                    : adresse premier trou (0 = pas de trou)
0000                    : libre (utilisé par les grilles)
0006                    : longueur du titre : 6 caractères
0000                    : numéro d'attribut vidéo (0 = pas d'attribut)
0002                    : numéro de ligne
001E                    : numéro de colonne
000F                    : adresse du texte formant le titre
53C5                    : S E - texte
44D2                    : D R - du
C9C3                    : I C - titre
0001                    : longueur de la variable réponse
0000                    : numéro de l'attribut vidéo
0008                    : numéro de ligne
0028                    : numéro de colonne
0020                    : adresse menu fils 1
0028                    : adresse menu fils 2
4DC5                    : M E - nom
4E55                    : N U - du menu
4DC3                    : M C - Fils 1
B1B1                    : 1 1 - (choix 1)
0000                    : le
0000                    : choix 2
0000                    : est un choix
0000                    : terminal
0006                    : longueur du 1er item
0000                    : attribut vidéo
0004                    : numéro de ligne du 1er item
000A                    : numéro de colonne
0025                    : adresse du libellé
C348                    : C H - libellé
CFC9                    : O 1 - de
D8B1                    : X 1 - l'item 1
0006                    : description
0000                    : du
0006                    : deuxième
000A                    : item
002D                    : .
C348                    : .
CFC9                    : .
D8B2                    : .
```

5.3.3 Création et modification de menus

5.3.3.1 Création

Nous avons développés pour cela un module client spécifique.

L'intérêt était double :
Grâce aux fonctionnalités de SEDRIC nous assistions l'utilisateur dans sa tâche de concepteur de menus;
C'était également une façon de valider ces fonctionnalités.

Néanmoins, en raisons du délai imparti pour la réalisation de la maquette, la puissance de ce module client est volontairement limitée.

L'algorithme mis en place permet de décrire plus qu'une simple arborescence puisqu'à un item, on peut faire correspondre un menu qui existe déjà : d'où la possibilité de passer d'une branche à l'autre, de boucler sur un même menu,...

La façon dont on utilise la pile de stockage des menus fait parcourir l'arbre de la manière suivante : (sur un exemple)

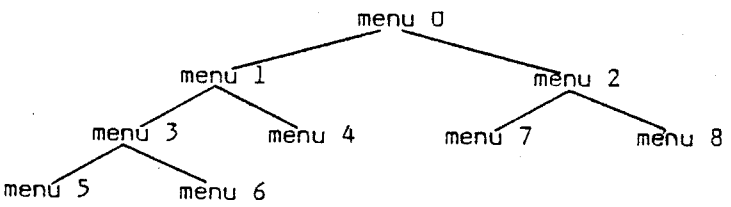


Fig. 22

L'opérateur a cité successivement l'existence des menus 0, puis 1 et 2, puis 3 et 4, puis 5 et 6, puis 7 et 8; les menus lui sont reproposés, pour être décrits et créés, dans l'ordre suivant : menu 0, 1, 3, 5, 6, 4, 2, 7, 8.

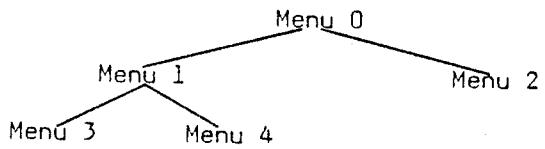
L'ensemble de l'algorithme a été écrit en Pascal.

Voici celui utilisé lors de la création de menus

```
Nbmenus=1; début=1;
Tant que nbmenus>0 faire
        Si début=1 faire
                Menu courant=menu général;
                Nbmenus=0; début=0;
        Sinon faire
                Menu courant=dernier menu stocké
                Nbmenus=nbmenus-1
        Fin si

        Donner le titre et les items du menu courant (Sedric)

        Pour chaque item faire
                . Donner le nom du menu correspondant à l'item
                (Sedric)
        Fin pour

        Ecrire l'article du fichier

        Pour chaque item faire
                Si le menu de l'item n'existe pas faire
                . Mettre en réserve ce menu
                . Nbmenus=nbmenus+1
                Fin si
        Fin pour
Fin tant que
```

Voici sur un exemple la description du mode opératoire.
Soit l'arbre de menus suivant à créer :

Fig. 23



Le dialogue a lieu comme suit : (en majuscules : caractères tapés par l'opérateur, en minuscules : messages envoyés par le module)

```
1   +  nom du menu général   :   MENU 0                    +

2   +  titre et items du menu  menu 0                      +
    +                    SEDRIC 1                          +
    +       ITEM 1                                         +
    +       ITEM 2                                         +

3   +  menus des items de menu 0                           +
    +  item 1      MENU 1                                  +
    +  item 2      MENU 2                                  +
    +                                                      +

4   +  titre et items du menu  menu 1                      +
    +                    SEDRIC 2                          +
    +   ITEM 3                                             +
    +         ITEM 4                                       +

5   +  menus des items de menu 1                           +
    +  item 3      MENU 3                                  +
    +  item 4      MENU 4                                  +
    +                                                      +

6   +  titre et items du menu  menu 3                      +
    +         etc...
```

Fig. 24

5.3.3.2 Modification de menus

L'usager ne désire pas toujours créer un menu mais parfois modifier ceux qui existent déjà et ceci d'une façon aisée.
Il nous a semblé bon d'offrir à l'usager une façon d'opérer qui soit homogéne avec celle qu'il a utilisée pour créer des menus.
La solution consiste alors à lire l'article désiré, le présenter à l'opérateur pour qu'il lui fasse subir les modifications voulues.
L'extension apporter au module client chargé de la création consiste à implémenter ce qui suit :

- si le menu courant n'existe pas, il n'y a pas de changement;

- si le menu courant existe déjà, au lieu d'afficher des blancs modifiables, on affiche le contenu actuel du menu, en modifiable; ainsi, deux cas se présentent :

. L'opérateur ne veut pas modifier le menu : dans ce cas il ne touche à rien et le menu est réécrit tel quel;

. L'opérateur veut modifier le menu : il lui suffit de taper des blancs par dessus une chaîne de caractères à supprimer (ou d'appuyer sur la touche 'CHAR-DEL'), de taper des caractères là où il veut créer un nouvel item, etc...

Ceci permet de considérer la création et la modification de menus comme un même processus.

5.4 ARCHITECTURE

Nous allons présenter dans cette partie l'architecture logicielle mise en place pour réaliser notre maquette.

5.4.1 Présentation

Nous rappelons que l'un des buts recherchés est de concevoir un serveur utilisable avec différents types de terminaux, depuis le "smart terminal" jusqu'au terminal intelligent. Il nous fallait donc adopter une architecture facilitant le départ de certaines fonctions. Il était, pour cela, indispensable de développer un logiciel très modulaire et structurée de façon hiérarchisée, donc en couches.
On remarquera sur la figure 25 la symétrie de cette architecture qui est le fruit de la volonté d 'offrir un temps de réponse acceptable.
Les tâches se séparent en deux groupes; l'un traite des appels venant des modules clients ou d'autres tâches à destination du périphérique; l'autre est chargé, parallèlement, des messages provenant de la console. Ainsi il y a simultanéité de traitement des messages "entrants" et "sortants".

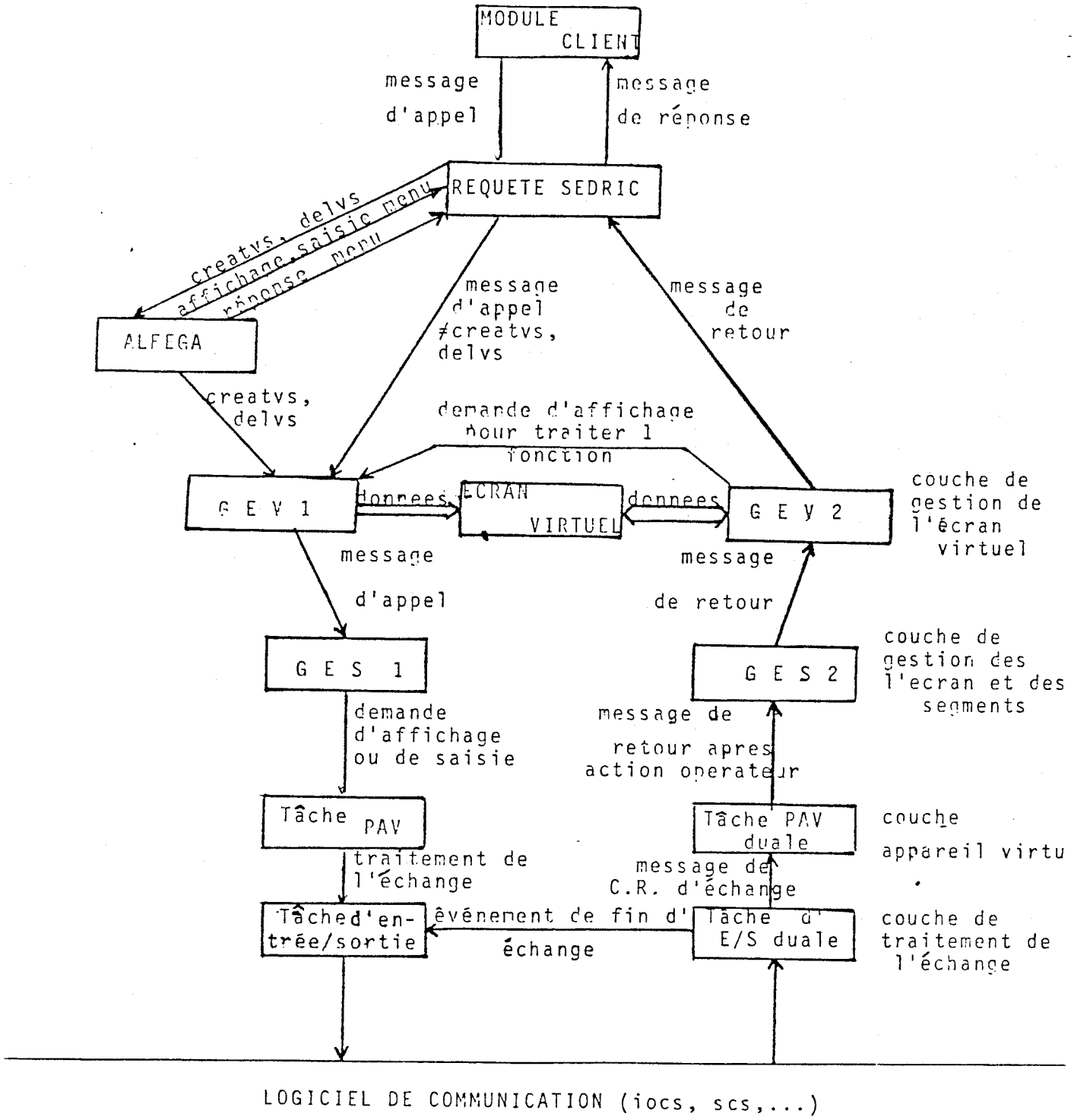

MODULE CLIENT
message d'appel
message de réponse
REQUETE SEDRIC
creatvs, delvs
affichage saisie menu
réponse menu
ALFEGA
message d'appel ≠creatvs, delvs
message de retour
creatvs, delvs
demande d'affichage pour traiter la fonction
G E V 1
données
ECRAN VIRTUEL
données
G E V 2
couche de gestion de l'écran virtuel
message d'appel
message de retour
G E S 1
G E S 2
couche de gestion des l'ecran et des segments
demande d'affichage ou de saisie
message de retour apres action operateur
Tâche PAV
Tâche PAV duale
couche appareil virtu
traitement de l'échange
message de C.R. d'échange
Tâched'entrée/sortie
événement de fin d'échange
Tâche d' E/S duale
couche de traitement de l'échange
LOGICIEL DE COMMUNICATION (iocs, scs,...)

Fig. 25

5.4.2 Description fonctionnelle des différents modules

5.4.2.1 La requête SEDRIC

C'est le point d'entrée dans SEDRIC. Elle est adressée à l'aide d'une instruction "SVC" exécutée par l'interface langage pour le compte du module client. Cette interface est en fait un sous-programme qui doit être link-édité avec le code appelant.
Les paramètres d'appel sont :

P1 : adresse du message d'appel

P2 : adresse du message de retour

P3 : adresse du compte rendu d'appel

Notons que le compte rendu se trouve également dans le message de retour

L'interface réalise l'appel ainsi :

```
RA := P1;
RB := P2;
SVC (SEDRIC); <SEDRIC SVC DE NUMERO 100;
```

Au niveau du module client, il y a deux types d'appel; l'un en PL16 qui prend la forme suivant :

```
CALL SEDRIC ( MSGAPP, MSGRET, CRD);

Avec en partie déclaration :
EXT PROCEDURE SEDRIC;
```

L'autre en PASCAL qui se fait ainsi :

```
SEDRIC (MSGAPR, MSGRET, CRD);

Avec en partie déclaration :
PROCEDURE SEDRIC (VAR MSGAPP : MSG, VAR MSGRET : MSG,
                  VAR CRD : INTEGER);
```

MSG étant un type structuré

Statut de la requête

Elle se déroule en mode maître (mode moniteur), sous la priorité du module client qui est une tâche, au sens RTESD.C' est un processeur "Foreground".

Mode de retour

On distingue deux modes qui dépendent des primitives. Celles-ci sont en effet de deux types :

- les unes sollicitent l'opérateur, ce sont

```
CREATVS, s'il y a un menu, et
RECEIVE
```

- les autres servent plutôt à l'informer ou a fournir des données au module client en ne nécessitant par l'intervention de l'usager, ce sont :

```
DISP,
READVS,
VALFONC,
```

DELVS.

Dans le premier cas le module client est suspendu. Pour cela SEDRIC exécute un 'WAIT' sur un sémaphore paramétré par le numéro d'écran virtuel. En retour, les tâches ALFEGA et GEV1 ou GEV2 font SVC (SEDRIC) en fournissant, entre autre, le numéro d'écran virtuel. La requête exécute alors un 'ACT' sur le même sémaphore. Ainsi le module client est réactivé.

On peut schématisé cela ainsi :

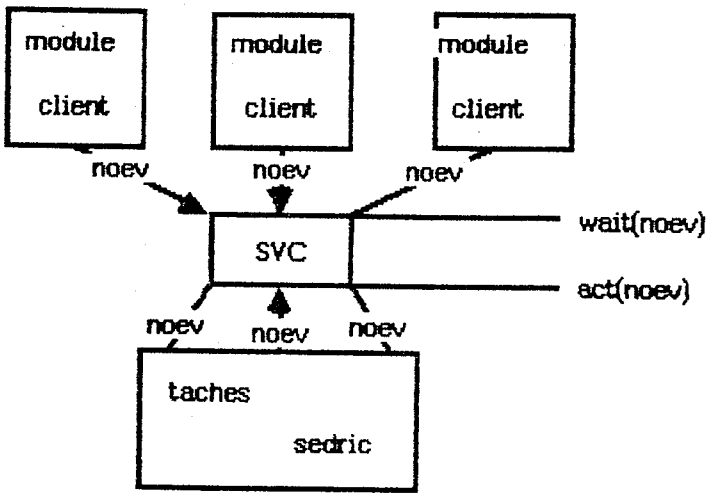


Fig. 26

5.4.2.2 La tâche ALFEGA (ALPHA et OMEGA)

Elle est activée àla suite d'une primitive CREATVS ou DELVS ou bien par la touche MENU-HISTORIQUE

C'est elle qui gère les menus et les historiques. Tous en étant une tâche du serveur de dialogue, elle utilise les services de SEDRLC comme si elle était un module client. Néanmoins, elle n'est jamais suspendue. Dans le cas contraire,elle serait la plus part du temps en attente de la réponse de l'opérateur et ne pourrait pas traiter les messages parvenant des autres modules clients.

5.4.2.3 Les tâches GEV1 et GEV2 (Gestion des Ecrans Virtuels)

La tâche GEV1 est sollicitée à la suite d'une primitive DISP ou READVS. Elle remplit l'écran virtuel en fonction des informations passées dans le message d'appel. Elle construit le message nécessaire à GES1. GEV1 peut également être activée par suite d'une action de l'opérateur sur une touche fonction dont la réponse réclame l'affichage de données se trouvant dans l'écran virtuel. C'est le cas de :

- Sélection de segments,
- Window up et window down,
- enlargement,
- Displacement,
- Diminution,
- Division.

Elle assure également la correspondance entre la fenêtre virturelle et l'écran physique.

La tâche GEV2 met à jour l'écran virtuel pour prendre en compte les données

communiquées par l'opérateur lorsqu'il appuie sur une touche fonction. C'est elle qui prépare et fournit à GEV1 les messages servant à répondre aux fonctions citées plus haut.

5.4.2.4 Les tâches GES1 et GES2(GEstion des Segments d'écran)

Ce sont elles qui s'occupent de la gestion de l'écran physique. Elles sont responsables de l'allocation des segments et de leur agencement au fur et mesure des besoins (recouvrement, déplacement, réaffichage). Ce sont elles qui assurent la correspondance (le mutliplexage) entre segments d'écran et écrans physiques.
Tandis que GES1 conçoit le message d'appel de la tâche PAV en fonction de celui fourni par GEV1 et de son contexte d'exécution; GES2, elle, traite une demande de l'opérateur en transmettant un message à GEV2. Elle assure également la correspondance entre la fenêtre physique et l'écran virtuel associé.

5.4.2.5 Les tâches PAV (Protocole d'Appareil Virtuel)

Le rôle de la tâche PAV est de transformer le message reçu depuis GES1 en un message exploitable par la console de visualisation. Elle analyse plus particulièrement les entêtes de blocs pour les traduire en des suites de codes ASCIi conformes à la logique du terminal.

Par exemple un positionnement en ligne 3 colonne 4 sera traduit par les codes :

27, 189, 35, 36 pour une console "Télévidéo"

Inversement, la tâche PAV duale fera la transformation dans l'autre sens pour fournir à la tâche GES2 un message conforme aux règles de SEDRIC.

Cette fonctionnalité est assurée grâce à des tables qui contiennent autant de points d'entrée qu'il y a de fonctions réalisées par SEDRIC sur le terminal (ex : clear, positionnement, verrouillage de lignes, etc...)
Ces tables, qui dépendent, du type de terminal conduisent à la séquence de code associée à la fonction demandée. Elles sont facilement extensibles
Il est prévu également de régler certains cas spécifiques du terminal par l'appel de sous-programme.
Ces deux tâches confèrent à SEDRIC une très grande souplesse qui lui permet de gérer réellement un parc de terminaux hétérogène.

5.5 LA COMMUNICATION INTERTACHES

Le mode de communication implémenté entre SEDRIC et les modules clients a été généralisé au niveau des tâches du serveur. En effet, elles communiquent entre elles par le biais d'un message et suivant un schéma producteur-consommateur. Les différents modules de SEDRIC disposent d'un contexte qui assure leur réentrance. ALFEGA, GEV et la requête SEDRIC ont un contexte par écran virtuel. GES, PAV et les tâches d'entrée-sortie disposent d'un contexte par console. Le contexte de GES et GEV contient un sous-contexte par fenêtre. Pour chaque tâche duale le contexte est commun. Cette zone de réentrance contient pour tous les modules les données propres à l'écran virtuel ou à l'écran physique, éventuellement la description des fenêtres virtuelles ou physiques, ainsi que le contexte d'exécution.
Une tâche désireuse d'envoyer un message précise le numéro de la tâche consommatrice ainsi que le numéro du contexte destinaire. A l'aide de ces informations, deux "SVC" (SEND et RECEIVE), de numéro 101 et 102 effectuent ce transfert. Elles utilisent également une zone de mémoire tampon gérée dynamiquement (procédures GETMEM, FREMEM). En outre, chaque tâche dispose d'une file d'attente par contexte depuis laquelle sont retirés les messages déposés par d'autres tâches. Ces files d'attente sont gérées en FIFO. Un tableau de sémaphore permet de synchroniser producteurs et consommateurs. On peut résumer cela par les schémas suivants :

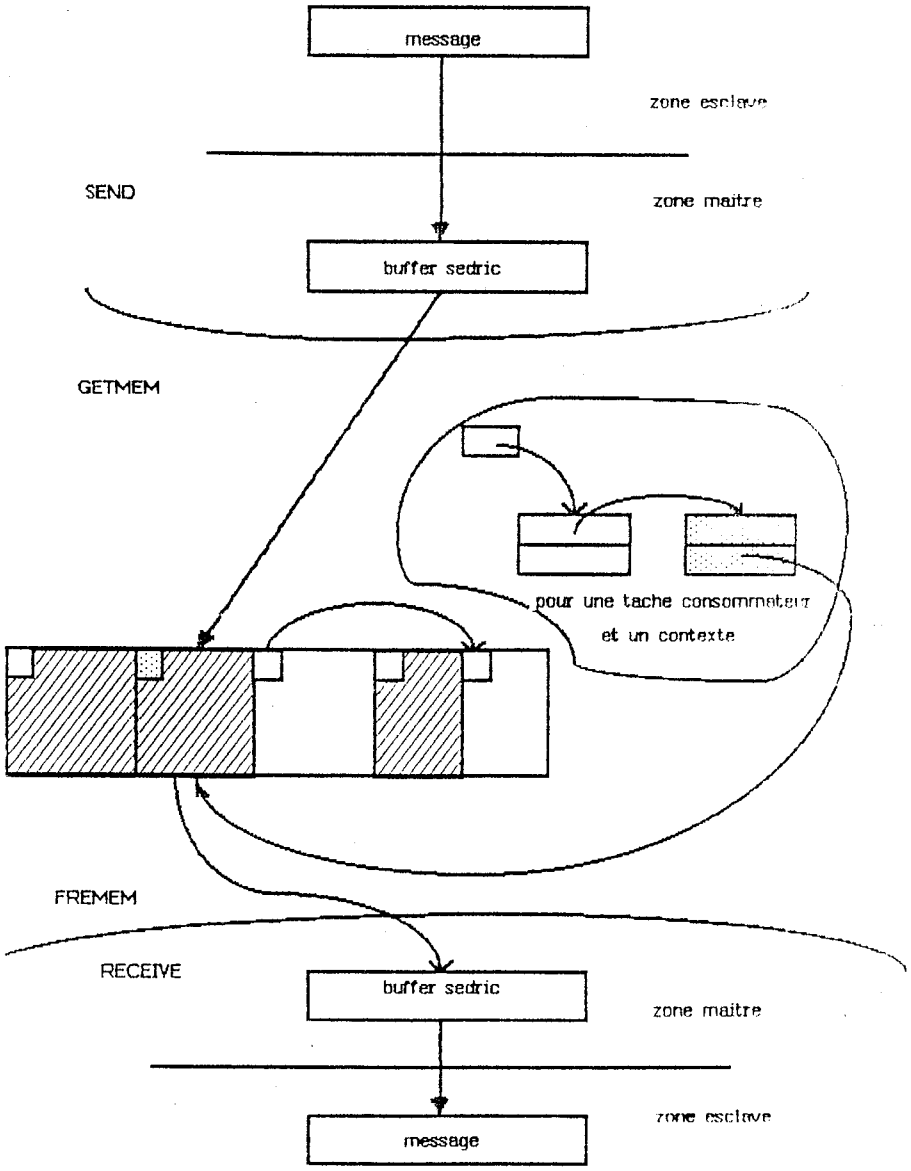


Fig. 27

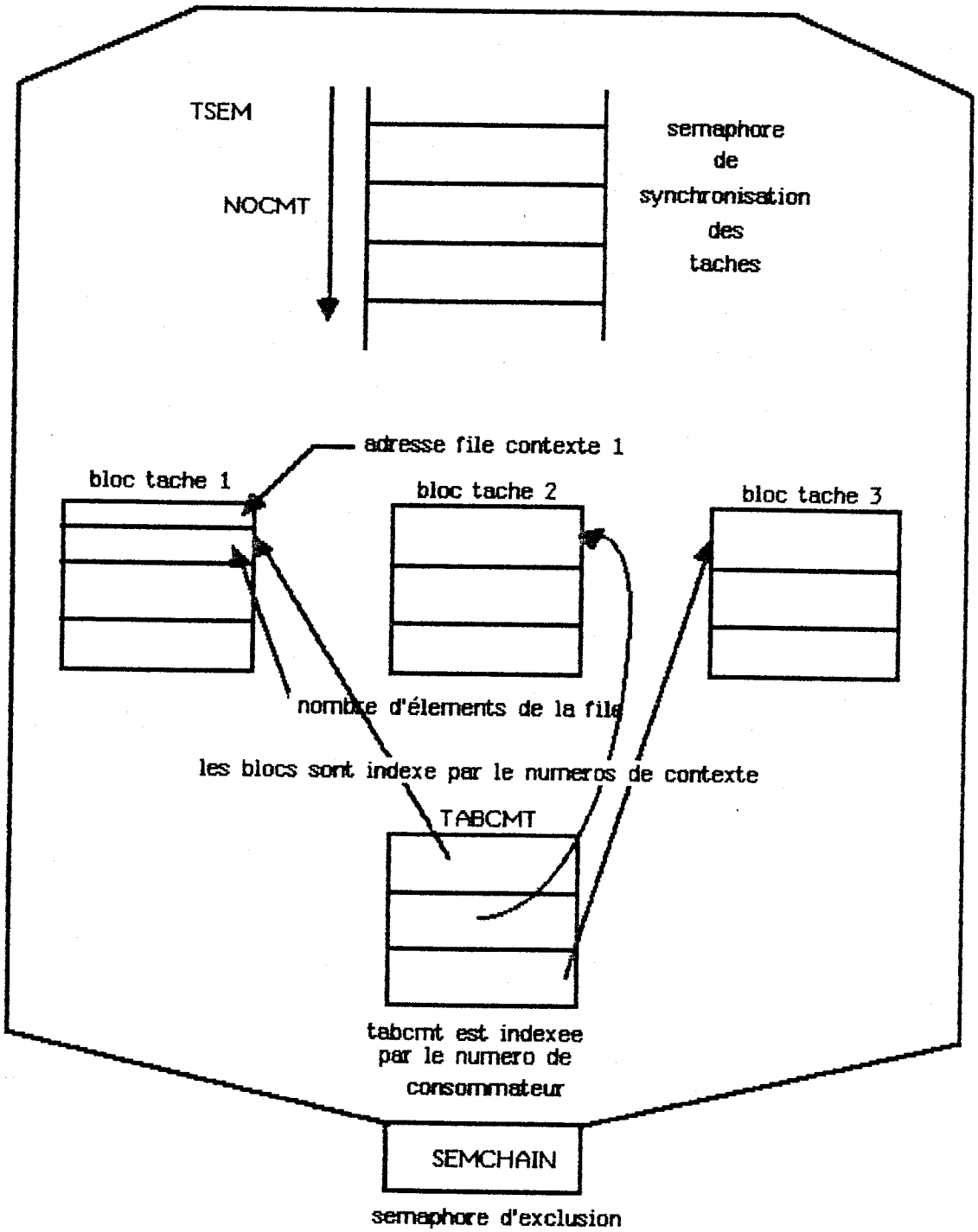

TSEM
NOCMT
semaphore
de
synchronisation
des
taches
adresse file contexte 1
bloc tache 1
bloc tache 2
bloc tache 3
nombre d'élements de la file
les blocs sont indexe par le numeros de contexte
TABCMT
tabcmt est indexee
par le numero de
consommateur
SEMCHAIN
semaphore d'exclusion

Fig. 28

5.5.1 Justification de ce mode de communication

- Il est homogène et peut être gérer de façon centralisée et donc optimale,

- Les messages étant stockés dans des files d'attente gérées en FIFO. La priorité des modules clients et des différentes tâches de SEDRIC n'a pas d'incidence pernicieuse sur la prise en compte de ces messages.
Ceci permet de donner à l'opérateur une relative indépendance par rapport au module client. En effet, chaque couche a la possibilité de se dérouler en asynchronisme avec les couches voisines. Si bien que, l'implémentation est telle que la couche GEV dépend totalement des sollicitations du module client alors que les couches en aval sont, elles, dépendantes de l'opérateur. La butée est assuréepar l'écran virtuel. Cet asynchronisme n'est pas respecté pour les primi tives RECEIVE et CREATVS avec menu puisqu'il faut attendre la réponse de l'opérateur pour la fournir au module client.

L'intérêt du message pour passer des paramètres a quant à lui, été développé dans le chapitre "interface logiciel".

5.5.2 La gestion des files d'attente et de la mémoire

Les files d'attente

Ce sont des files à double chaînage. On déchaine le dernier et on chaine en tête. Les éléments sont en fait des adresses de buffer dans la mémoire dynamique.

La mémoire

Son adresse d'implémentation et sa taille sont configurables sur le site. Elles sont laissées à l'appréciation de l'usager. Il faut noter, toutefois, que la saturation de la mémoire dynamique est une cause de sévère dégradation des performances. En standard deux K mots de 16 bits sont réservés dans la requête.

Le principe est le suivant : un mot, commun aux sous-programmes GETMEM et FREMEM contient l'adresse d'implantation de cette mémoire. A l'origine elle est entièrement libre. De ce fait le premier mot de cette mémoire contient l'adresse du dernier mot qui vaut 'nil' (-1); le deuxième mot contient la taille. Il n'y a à ce stade qu'un boc libre. Si une tâche désire envoyer un message d'une longueur de 100 mots, ils seront pris en fin de mémoire. Il n'y aura toujours qu'un seul bloc libre dont la taille aura varié en conséquence. Ce schéma se perpetuera tant que seules des tâches productrices enverront des messages. Cependant les files d'attentes associées aux tâches destinatrices sont mises à jour en fonction des contextes concernés (écrans virtuels ou physiques). Les éléments de ces files pointent sur les messages rangés dans la mémoire dynamique. La consommation de ces messages se fera alors dans une chronologie qui ne fut pas forcément celle de leur production. Cela provoque inévitablement la création de trous en mémoire qui vont être chaînés entre-eux. L'allocation de la mémoire se fera en fonction de ces blocs libres, de leur chainage, de leurs tailles et de celle des messagesenvoyés.

5.5.3 Algorithme de principe d'un module SEDRIC

Notons tout d'abord que le sous-programme RECEIV, outre le fait qu'il fournit un message, indique à la tâche consommatrice, à l'aide d'un booléen, appelé, ici, 'YAMSG', la présence d'au moins un message dans les files d'attente de la tâche. S'il n'y a aucun message ce sous-programme, implémenté sous forme de SVC s'exécutant avec la priorité de la tâche, la bloque sur l'un des sémaphores du tableau que nous avons présenté plus haut (TSEM). Ce tableau est indicé par le numéro de consommateur.

```
DECLARATION-DE-LA-TACHE-DUALE  (SI TACHE PRINCIPALE)

INITIALISATION DE L'ADRESSE DES CONTEXTES

NUMERO-CONTEXTE:=0

FAIRE;
|
|     YAMSG:=FAUX;
|
|     TANT-QUE (NON YAMSG)
|     |
|     |    CALCUL-NUMERO-CONTEXTE-SUIVANT
|     |
|     |    ADRESSAGE CONTEXTE
|     |
|     |    RECEIV(NOCONSOSOURCE,
|     |
|     |           MSGCONSO,
|     |
|     |           NOCTXSOURCE,
|     |
|     |           YAMSG);
|     |
|     FIN-TANT-QUE
|
|     TRAITEMENT DU MESSAGE RECU
|
|     SEND(NOCONSODESTINATAIRE,
|
|          MSGPRODUITS
|
|          NOCTXDESTINATION);
|
FIN-FAIRE;
```

On remarque donc que les tâches SEDRIC bouclent sur l'acquisition d'un message.

Voici les algorithmes de SEND et RECEIVE

SEND

1. Appropriation du sémaphore d'exclusion
2. Demande de mémoire pour le message.
3. Rangement du message dans la mémoire dynamique.
4. Chaînage du message dans la file correspondant au numéro de contexte
5. Décrémentation du compteur d'éléments de la file
6. RLSE sur le sémaphore du consommateur pour signaler l'arrivée du message et éventuellement libérer une tâche bloquée.
7. Libération du sémaphore d'exclusion

RECEIVE

1. RQST sur le sémaphore de synchronisation du consommateur. Il est bloqué s'il n'y a aucun message pour lui sur les différents contextes
2. Acquisition du sémaphore d'exclusion
3. Calcul de l'adresse de la file en fonction du numéro de consommateur
4. Si la chaine n'est pas vide

   Alors

   4.1 Déchaine le message.

   4.2 Transfert du message chez l'appelant.

   4.3 Libération de la mémoire

   4.4 booléen := vrai.

5. Sinon

   5.1 Booléen := faux Il faut que l'appelant demande 1 msg sur 1 autre contexte

   Fsi

6. Libération du sémaphore d'exclusion.

5.6 LA GESTION DES ENTREES-SORTIES

5.6.1 Les contraintes de performances

Un échange sur une console peut prendre un temps prohibitif. Le délai pour une entrée dépend, dans la limite du time-out, de l'opérateur. Un affichage peut pendre un temps qui n'est pas négligeable. Un écran complet fait 1920 caractères. Un caractère est codé sur 10 bits. La vitesse de transmission étant rarement supérieure à 4800 bands sur SOLAR, l'affichage d'un écran plein dure 4 secondes.

Il est donc hors de question que la tâche qui lance l'échange soit suspendue pendant toute sa durée.

Celle-ci pourraitl'initialiser, faire un traitement puis se mettre en attente de la fin d'échange.
Cependant cette programmation n'est pas optimale.
En effet, la fin d'échange peut survenir avant que le module l'attende ou bien trop tard. Il serait bien plus intéressant que la tâche lançant l'entrée-sortie puisse continuer son traitement. L'événement de fin d'échange serait alors signalé à une autre tâche chargée exclusivement de ce traitement. C'est ce que réalise le mode 'XMOD' sur IOCS

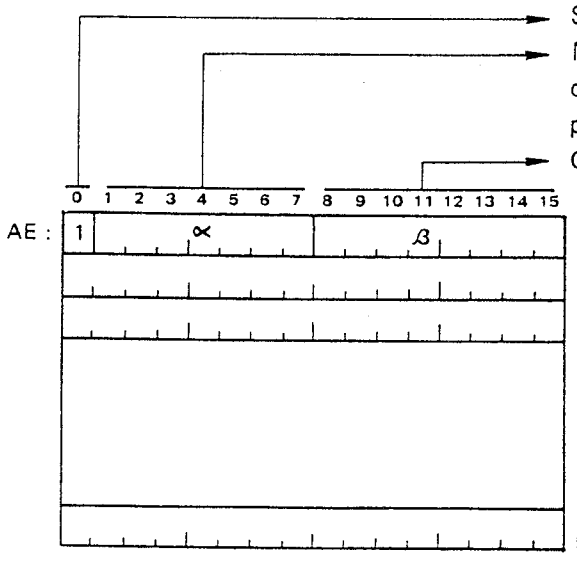
5.6.2 L'échange en 'XMOD'

Nous nous devons de faire ici quelques précisions au sujet de l'implémentation des sémaphores sur le SOLAR.

Il y a deux type de sémaphores qui sont :
- les sémaphores d'exclusion
- les sémaphores privés.

Ce sont eux qui sont utilisés ici.

En voici la description schématique :



F : file des indicateurs de paramètres, permettant de personnaliser l'exécution de la tâche software de numéro ∝, par le passage d'un paramètre fonction du rang du bit indicateur dans la file

Fig. 29

Deux instructions agissent sur un sémaphore privé.
Ce sont :

. WAIT : mise en attente sur le sémaphore

- ACT : activation d'une éventuelle tâche en attente sur le sémaphore

Dans le cas d'un sémaphore privé avec paramètre celui-ci est fourni dans le registre 'Y'.

Un échange en XMOD a les caractéristiques suivantes :

Il n'est utilisable que par une tâche se déroulant en mode maitre.

Il permet de réveiller une tâche en attente sur un sémaphore paramètré. A la fin de l'échange, IOCS (le moniteur d'entrée-sortie SOLAR) exécute une instruction ACT sur le sémaphore avec comme paramètre le numéro de la console où l'échange s'est terminé.

La tâche lancant l'échange fournit dans un bloc d'appel les paramètres necessaires et notamment l'adresse du sémaphore.

5.6.3 Utilisation dans SEDRIC

Elle se fait au niveau des tâches d'entrées-sorties que nous appelons TI01 et TI02 pour la tâche duale.

L'algorithme de principe de TIO1, qui ressemble à celui que nous avons déjà vu, se résume trés schématiquement de la façon suivante :

```
DECLARATION-DE LA TACHE-DUALE

INITIALISATION DE L'ADRESSE DES CONTEXTES

NUMERO-CONTEXTE:=0

FAIRE;
|
|     YAMSG:=FAUX;
|
|     TANT-QUE (NON YAMSG)
|     |
|     |    CALCUL-NUMERO-CONTEXTE-SUIVANT
|     |
|     |    ADRESSAGE CONTEXTE
|     |
|     |    RECEIV(NOCONSOSOURCE,
|     |
|     |           MSGCONSO,
|     |
|     |           NOCTXSOURCE,
|     |
|     |           YAMSG);
|     |
|     FIN-TANT-QUE
|
|     TRAITEMENT DU MESSAGE RECU
|
|     APPEL D'IOCS (XMOD,ADRESBUFFER,NOCONSOLE,...)
|
|     ( ON N'EST PAS SUSPENDU
|
|       ON CONTINUE EN SEQUENCE)
|
FIN-FAIRE;
```

Pour la tâche TI02, il est très schématiquement le suivant :

```
FAIRE;

        WAIT (ADRESS-SEMAPHORE)

        RECHERCHE DU NUMERO DE CONSOLE

        ( 1ER BIT A UN DANS LA FILE DU SEMAPHORE )

        ADRESSAGE DU CONTEXTE D'APRES CE NUMERO

        Si ECHANGE = SORTIE alors

                                TRAITEMENT 1

                Sinon

                                TRAITEMENT 2

        Fin-Si


        SEND (PAV2,
                MESSAGE,
                NUMERO-CONTEXTE);

FIN-FAIRE;
```

5.6.4 La gestion du break

Le 'break' joue un rôle prépondérant puisqu'il conditionne le changement de segment. Sa prise en compte doit être immédiate.

En standard le driver signale le break sans délai s'il intervient lors d'une réception : c'est un défaut qui interrompt l'échange.

Par contre, il est ignoré lors d'un affichage. Il ne sera traité qu'une fois l'échange terminé. Une réception est alors initialisée pour traiter les caractères parasites survenant entre deux échanges. Le 'break' peut alors être pris en compte.

Il est clair que ce n'est pas suffisant.

C'est pourquoi IOCS permet d'appeler un sous-programme utilisateur dont on a donné l'adresse par ailleurs. Il est appelé directement par le driver lors de l'apparition du 'break'.

Application à SEDRIC

Dans sa phase dinitialisation, TI01 déclare l'adresse du sous-programme à IOCS. Le sous-programme implémenté dans TI02 se déroule indépendamment de cette tâche. En effet, étant directement appelé par le driver il s'exécute sous sa priorité.

5.6.5 Explicitation du PAV

Voici schématiquement les tables utilisées.

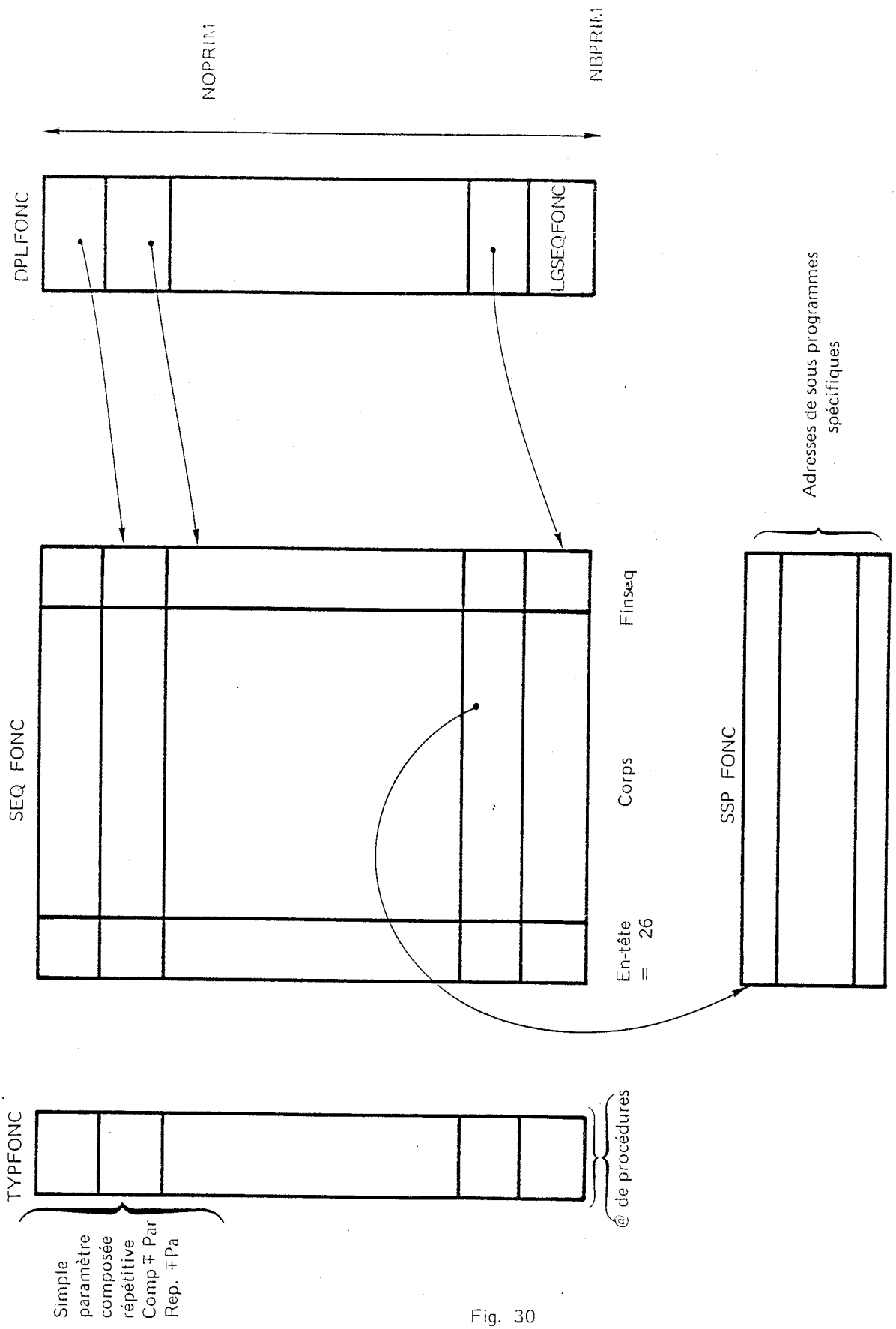


Fig. 30

Table SEQFONC

L'emplacement 'i' contient les codes pour exécuter la fonction 'i'.

Exemple :

Fonction zéro soit 'clear en demi intensité'.
L'emplacement zéro contient
27, "'", 27, ",".

L'exécution de la primitive demandée nécessite parfois la connaissance de certains paramètres.
Exemple :

Positionnement du curseur. Les paramètres sont dans ce cas les numéros de ligne et de colonne.

Numéro de primitive : 3

SEQFONC (3) = 27, "=", 27, 0, 27, 0

Il faut ici connaitre les numéros de ligne et de colonne. L'entête (code 27) répétée dans le corps de la séquence signale l'adjonction d'un paramètre à l'endroit précis où il se trouve. Ceci est validé par le tableau TYPFONC. En effet TYPFONC (3) = Fonction paramétrée.

Rôle de TYPFONC :

Il fournit le type de chaque fonction. TYPFONC (i) = adresse du sous-programme gérant ce type de fonction.
Ce dernier exploite les tableaux SEQFONC et SSPFONC.
En effet, le code situé juste après 27 est l'index dans SSPFONC pointant sur l'adresse d'un sous-programme traitant spécifiquement la fonction.
Dans le cas présent cet index vaut 0. Ce sous-programme est appelé deux fois. Une fois pour le numéro de ligne une autre fois pour le numéro de colonne.

Le tablau DPLFONC

Il indique le début de chaque séquence dans SEQFONC.
DPLFONC (i) = index dans SEQFONC pointant sur l'entête de la fonction i.

Exemple : DLPFONC (2) = 7.

Remarques :

Ces tables dépendent, du type de la console. Elles sont donc répétées en conséquences. Ce qui a été présenté ici concerne la 'télévidéo' puisque c'est elle qui équipe la maquette.

5.7 LA GESTION DES ECRANS VIRTUELS

Nous utilisons 5 tableaux :

EV

EVATT

SUCCEV

PREDEV

POSEV

EV :

C'est un tableau de 80 colonnes et d'un nombre de lignes donné par la commande CREATVS. Il contient tous les caractères qui sont successivement apparus sur l'écran. Il est structuré en lignes.

EVATT :

NBATTLIG * 2

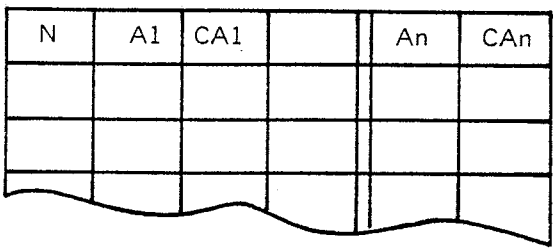


Fig. 31

Tableau de (NBATTLIG+1) X TAILEV mots contient pour chaque ligne de l'EV le nombre d'attributs de cette ligne (N) et la description des N attributs.

Ai= code du iième attribut de la ligne,
CAi= abscisse de l'attribut i dans la ligne.

Les tableaux suivants ont été implémentés pour soulager un compositeur de texte des fonctions d'ajoût, de suppression ou de déplacement de lignes.
Bien qu'étant gérés dans la maquette, ils n'y ont pas de justification propre.
Le terme 'TAILEV' indique le nombre de lignes de l'écran virtuel.

SUCCEV

Tableau de TAILEV mots
SUCEV(i) = numéro de la ligne suivant la ligne i. (index dans EV)
Au début SUCCEV(i) = i+1

PREDEV

Tableau réciproque de SUCCEV. Sa longueur est TAILEV.
PREDEV(i) = numéro de la ligne précédant la ligne i (index dans EV).
A l'initialisation PREDEV(i) = i-1

POSEV

Tableau de TAILEV mots
POSEV(i) donne l'indice dans EV de la ligne de numéro i.
A l'initialisation POSEV(i) = i.

Ces trois derniers tableaux sont résidents en mémoire. EV et EVATT ont une partie en mémoire : celle concernant la fenêtre virtuelle, l'autre partie sur disque dans des fichiers à accès directs par le numéro de lignes.

Tous ces tableaux sont gérés en mode rouleau.

5.8 GESTION DES FENETRES MULTIPLES

5.8.1 Les fenêtres dans SEDRIC

La console qui nous a été imposée rend presque impossible l'implémentation des fenêtres autrement que sous la forme d'une suite de lignes entières et contigues. Il faudrait faire un contrôle permanent du curseur et c'est incompatible avec le mode de transmission utilisé : le mode bloc.

Les fenêtres peuvent être :

Contigues

Partiellement recouvertes ou

Totalement recouvertes.

5.8.2 L'allocation des fenêtres physiques

L'algorithme d'allocation se situe dans GES. En voici le principe :

- On essaie d'allouer en totalité des lignes libres sur l'écran.
- S'il n'y en a pas suffisamment on essaie de trouver la plus longue suite de lignes vides au-dessus ou en-dessous du dernier segment utilisé de façon à l'épargner le plus possible.

5.8.3 Les fenêtres multiples

Le rôle de GEV

Le multiplexage des fenêtres est du ressort des deux couches GEV et GES.
La première d'entre-elles y contribue fortement car elle assure la multiplicité des écrans virtuels. Elle associe un contexte d'exécution lors de la création de chacun d'entre eux.
Or à chaque écran virtuel correspond une seule fenêtre virtuelle et un seul segment d'écran.
Il faut néanmoins cons idérer le cas particulier du dédoublement de fenêtres qui fait exception. En effet à un écran virtuel correspondent deux fenêtres virtuelles et deux fenêtres physiques. Cependant seule l'une d'elles est courante à un instant précis. La correspondance biunivoque entre les trois entités est donc respectée.

Le rôle de l'écran virtuel

Il conditionne le recouvrement des fenêtres physiques. En mémorisant leur contenu depuis la création jusqu'à la suppression, c'est lui qui permet de les restituer lors d'une sélection.

Le numéro de segment

C'est grâce à lui que se fait la correspondance entre la fenêtre virtuelle et le segment d'écran.
Il est constitué du couple :

Numéro d'écran virtuel (CF CREATVS),

Numéro de fenêtre dans l'écran (pour gérer le dédoublement de fenêtres).

Il est indiqué dans chaque message que la tâche GEV1 envoie à la GES1 et réciproquement dans chacun que la tâche GEV2 reçoit de GES2.

Rôle de la couche GES

Elle connait à chaque instant la topologie de l'écran puisque c'est elle qui en contrôle l'agencement. Elle est donc à même d'afficher dans le segment adéquat le contenu de la fenêtre virtuelle transmise par GEV1. Réciproquement, elle fournit, en retour, à GEV2 les informations provenant de l'opérateur, en spécifiant par le numéro de segment, la fenêtre et l'écran virtuel concernés.

5.8.4 Les structures de données utilisées par GES

Remarque :

Une ligne de l'écran est dite occupée si elle contient au moins un caractère appartenant à une fenêtre physique créée et non encore détruite (voir CREATVS et DELVS).

TFREELIG : tableau de 24 booléens (taille de l'écran) indiquant si une ligne est libre ou occupée.

TFIRSTLIG : TFRISTLIG (i) = numéro de la première ligne du segment "i".

TLASTLIG : TLASTLIG (i) = numéro de la dernière ligne du segment "i".

TFIRSTCOL : idem TFIRSLIG mais avec le numéro de colonne.

TLASTCOL : idem TLASTLIG mais avec le numéro de colonne.

Le couple (FIRSTLINE, FIRSTCOL) indique les coordonnées du coin haut gauche de la fenêtre physique courante, tandis que (LASTLINE, LASTCOL) représente les coordonnées du coin bas droit. On détermine ainsi les limites des fenêtres.

Remarque :
Les tableaux relatifs aux colonnes ne sont pas utilisés dans la maquette. Ils sont justifiés par une éventuelle extension.

5.9 LA GESTION DU MULTIACTIVITE

Le multiactivité, qui consiste à associer plusieurs modules clients simultanément à la même console est rendu possible grâce à plusieurs éléments :

Le système

Le choix d'un système multitâche, qui ne rattache pas la notion d'usager à la console, en établissant une relation bijective entre les deux, comme le font la plus part des systèmes temps partagé, facilite le multiactivité.

Le break

Sa gestion, telle qu'elle est faite dans SEDRIC, pemet de quitter instantanément et à tout moment une activité quelconque.

Identification d'un module client

Celui-ci n'est pas connu en tant que tel par SEDRIC mais uniquement à travers l'écran virtuel ou les écrans virtuels qu'il a créés. Ce sont eux qui permettent de l'identifier grâce à leur numéro (voir primitive CREATVS) qui sert de paramètre à un sémaphore. Sur RECEIVE ou sur CREATVS (avec menu) le module client est suspendu, il est réactivé grâce à ce sémaphore lors du retour de la réponse utilisateur.

Contexte d'exécution

Chaque module de SEDRIC est réentrant. Pour cela, il gère un contexte qui est associé à l'écran virtuel ou à l'écran physique comme nous l'avons déjà décrit dans le paragraphe "communication inter-tâche". Les couches GEV et GES gérant des sous contextes pour les fenêtres.

Contenu d'un contexte

Il se compose généralement de :

- message reçu et envoyé
- variables associées (relai, index)
- éventuellement tables des écrans virtuels
- éventuellement tables du protocole d'appareil virtuel
- variables associées à ces tables
- indicateurs d'états
- variable de travail
- bloc d'appels aux moniteurs FMS et IOCS.

5.10 GESTION DU MULTICONSOLE

Elle est gérée par les couches GES, PAV et TIO. Chacune d'elles associe un contexte d'exécution par console. Elle assure ainsi la réentrance pour chaque terminal.

Au niveau des autres couches, la console, en elle-même, n'a aucune importance. Les zones de réentrance sont attachées aux écrans virtuels. Que deux d'entre eux soient rattachés à la même ou à deux consoles différentes n'a aucune conséquence.

Nous allons expliquer dans la suite du chapitre comment ont été réalisés les primitives et l'interface usager.

Nous présenterons pour cela, l'algorithme de principe déroulé dans chaque module.

5.11 LES PRIMITIVES.

Nous allons, ici, suivre les différentes primitives dans le sens SEDRIC usager, le retour est explicité avec l'interface usager.

5.11.1 CREATVS

5.11.1.1 creatvs : requête SEDRIC

L'algorithme se résume à cela

- calcul de VSNUM

- adressage du contexte d'après VSNUM

- analyse du message de l'appelant pour tester la définition de nouveaux délimiteurs et mettre à jour ceux pris par défaut (délimiteur d'entête de bloc, de texte, de fin de texte, de fin de message)

- envoi du message à la tâche ALFEGA

- suspension du module client.

Remarques :

Chaque tâche explore cycliquement toutes ses files d'attente en fonction d'un numéro de contexte qui correspond pour ALFEGA, GEV1 et GEV2 au numéro d'écran virtuel et pour GES1, GES2, PAV1, PAV2, TIO1 et TIO2 au numéro de console. L'écran virtuel ou la console traités ne sont pas déduits du message reçu mais imposés par la tâche réceptrice. Cela lui permet d'être maître du séquencement des messages. En conséquence, quelque soit le type de la primitive traitée, on a toujours, en début de tâche, la séquence suivante :

Calcul du numéro de la file à traiter (numéro de contexte)
Adressage de ce contexte
Demande de message pour ce contexte.

5.11.1.2 Creatvs : tâche ALFEGA

Mise à jour éventuelle des délimiteurs

Si présence menu alors

Mémorisation de son nom

YAMENU := VRAI

Profondeur-arbre := 0

Fsi

Transmission à la tâche GEV1 du message du module client.

Si YAMENU alors

Lecture du menu

Demande à SEDRIC d'afficher le titre
(primitive DISP avec clear préalable)

Demande à SEDRIC d'afficher chaque item
(boucle avec la primitive DISP sans clear sur le nombre d'items)
Demande de saisie pour la réponse
(RECEIVE sans blocage : RCVMENU primitive interne à SEDRIC de numéro 11)

Sinon

Fabrication du message réponse

Retour à SEDRIC

Fsi

5.11.1.3 Creatvs, tâche GEV1

Analyse du message reçu pour récupérer :

- le numéro de console

- la taille de l'écran virtuel

- la taille du segment qui sert à dimensionner la fenêtre virtuelle

- les délimiteurs s'ils sont redéfinis.

Création du fichier écran virtuel. Son nom est formé des lettres "EV" concaténées au numéro de l'écran virtuel.

Initialisation de la partie résidente en mémoire. Elle correspond à la fenêtre logique.

Initialisation des tableaux :

SUCCEV	: tableau des lignes suivantes
PREDEV	: tableau des lignes précédentes
POSEV	: tableau indiquant la position dans l'écran virtuel d'une ligne dont on donne le numéro (cf description de l'écran virtuel paragraphe 5.76).

Fabrication du message transmis à la tâche GES1. Il comprend :

01 (indique que c'est une primitive CREATVS)

Nom de l'écran virtuel

Numéro du segment

Taille du segment.

Liste des délimiteurs.

Transfert du message à la tâche GES1 en le chaînant sur une liste choisie d'après le numéro de console.

5.11.1.4 Primitive CREATVS : tâche GES1

Cette primitive signifie pour GES1 :
Création d'un segment.

Voici comment se déroule le traitement :

1. Si premier appel pour cette console :
   Initialisation du contexte

2. Récupération dans le message reçu des paramètres spécifiés plus haut

3. Allocation du segment d'après l'algorithme décrit au paragraphe 5.8.2.

4. Transfert à la tâche PAV2 d'un message sollicitant l'affichage d'un cadre délimitant la fenêtre physique (fonction numéro 9).

5.11.1.5 Primitive CREATVS : tâche PAV1

Cette tâche connait en fait trois types de primitives qui sont :

- demande d'affichage
- demande d'entrée
- exécution d'une fonction (exemple :
  - positionnement de curseur
  - clear
  - verrouillage d'une ligne
  - dessin d' un cadre
  - etc...).

Chacune de ces fonctions est identifiée par un numéro.

Dans le cas présent la primitive CREATVS provoque :

- Initialisation du contexte de la console si c'est le premier appel pour cette console.

- traitement de la fonction numéro 9 soit dessin d'un cadre dont les coordonnées sont spécifiées dans le message.

Dans le cas de la console télévidéo, support de notre réalisation, il s'ensuit l'envoi, vers la tâche TI01, du message suivant.

- 0 (indique une sortie vers la visu)
- nombre de caractères à envoyer
- adresse relative du buffer
- registre SL0

Suite de codes

Adresse du buffer :
Elle est relative au contenu du registre SL0.

Expliquons sur un exemple comment cela fonctionne (les valeurs sont codées en hexadécimal) :

L'adresse absolue du buffer est 12380
La couche PAV se trouve dans une partition située en 12000
Le registre SL0 vaut donc 1200 soit 12000 divisé par 10 (hexadécimal)
L'adresse du buffer dans le message est donc : 380.

Suite de codes :
C'est celle-ci

27, "=", numéro de 1ère ligne, 79 (positionnement fin de ligne)

27, "G", 0 (fin d'attribut)

27, "=", numéro 1ère ligne, 0 (positionnement en début de ligne)

27, "G", 8 (attribut ligne soulignée).

On matérialise ainsi la ligne formant le haut du cadre, le bas est affiché par l'envoi de codes semblables. Seuls les positionnements changent.

Remarque :
Le nom de l'écran virtuel est affiché au sommet du cadre.

En exploitant le message reçu, elle construit, pour IOCS, le bloc d'appel suivant :

Mot 0 : registre SL0

Mot 1 code sortie XMOD + numéro de la console

Mot 2 : -1, il indique que l'adresse du buffer est relative à SL0

Mot 3 : adresse du buffer

Mot 4 : code fin de texte(entrée sur code d'arrêt)

Mot 5 : réservé au compte-rendu (initialisé à zéro)

Mot 6 : adresse du sémaphore servant à signaler l'évènement de fin d'échange.

5.11.2 Primitive 'DISP'

5.11.2.1 Primitive 'DISP', requête SEDRIC

Vérification de la validité du message.

Transfert de celui-ci à la tâche GEV1.

Retour au module client.

5.11.2.2 Primitive DISP, tâche GEV1

Analyse de l'entête du message pour allouer, ou non, une nouvelle fenêtre
Si oui :

Recopie de l'ancienne fenêtre dans le fichier direct alloué à l'écran virtuel.

Le texte de chaque bloc du message est recopié dans l'écran virtuel à un emplacement donné par le tableau POSEV et par les numéros de ligne et de colonne contenus dans l'entête du bloc.

Le tableau des attributs est rempli de même.

Le numéro de ligne est fourni par le module client relativement au début de l'écran virtuel.
GEV1 le rapporte au début de la fenêtre.

Le message, ainsi modifié, est transféré à la tâche GES1.

5.11.2.3 Primitive DISP, tâche GES1

Si l'entête du message contient une demande de clear précédant l'affichage alors

Envoi à la tâche PAV1 d'un message sollicitant l'exécution de la fonction numéro zéro (effacement du segment).

Les numéros de lignes figurant dans les entêtes de blocs sont modifiés afin qu'ils ne soient plus relatifs au début de la fenêtre mais de l'écran physique.

Pour chaque bloc,

Fabrication d'un message de demande d'affichage.

Transfert vers la tâche PAV1.

5.11.2.4 Primitive DISP, tâche PAV1

Analyse du message pour :

Extraire les coordonnées
Construire le message de positionnement induit
Transférer ce message à la tâche TI01.

Traiter de même l'attribut vidéo.

Si l'affichage demandé est modifiable, on mémorise les coordonnées, car, en retour on recevra le texte modifiant cette zone. On le transmettra à la tâche GES2 en spécifiant les coordonnées mémorisées.

On construit un message de positionnement avec ces coordonnées.
Il est transféré à la tâche TI01.

Le contenu du texte fait l'objet du transfert d'un autre message.

5.11.2.5 Primitive DISP, tâche TI01

Les messages reçus sont des demandes d'affichage. Ils se traduisent par une SCV IOCS avec comme bloc d'appel :

Mot 0 : registre SL0

MOT 1 : code sortie en XMOD + numéro de la console

Mot 2 : -1, il indique que l'adresse du buffer est relative à SL0

Mot 3 : adresse du buffer

Mot 4 : taille du buffer en octets

Mot 5 : réservé au compte-rendu (initialisé à zéro)

Mot 6 : adresse du sémaphore servant à signaler l'évènement de fin d'échange.

5.11.3 RECEIVE

5.11.3.1 Receive : requête SEDRIC

- contrôle de la validité du message reçu
- adressage du contexte
- transfert du message à la tâche GEV1
- suspension du module client. (wait SEMEV (vsnum) : la requête se déroule sous la priorité du module client).

5.11.3.2 Reveive : tâche GEV1

Notification d'une primitive 'receive'

Insertion du numéro de segment dans le message.

Transfert de ce dernier vers la tâche GES1.

5.11.3.3 Receive, tâche GES1

Récupération du numéro de segment dans le message reçu. Il est placé juste derrière le type de la primitive.

Chaînage du segment sur la console. En effet, d'autres messages 'receive' provenant de modules clients différents (chacun d'entre eux étant suspendu) peuvent arriver pour la même console. On chaîne donc les segments concernés pour pouvoir les identifier quand on traitera la réponse de l'opérateur.

Transfert du message à la tâche PAV1.

5.11.3.4 Receive, tâche PAV1

Le message fabriqué pour la tâche TIO1 est le suivant :

Mot 0 : entrée d'information depuis la console

Mot 1 : sans signification

Mot 2 : registre SL0

Mot 3 : adresse du buffer d'échange par rapport au registre SL0.

5.11.3.5 Receive, tâche TIO1

La primitive 'receive' se transforme endemande d'entrée depuis la console.

TIO1 construit le bloc d'appel à IOCS (IOCB) à l'aide du message reçu depuis PAV1, soit :

Mot 0 : registre SL0

Mot 1 : code entrée + numéro de la console

Mot 2 : -1, il indique que l'adresse du buffer est relative à SL0

Mot 3 : adresse du buffer

Mot 4 : code fin de texte (entrée sur code d'arrêt)

Mot 5 : réservé au compte-rendu (initialisé à zéro)

Mot 6 : adresse du sémaphore servant à signaler l'évènement de fin d'échange.

5.11.4 Primitive DELVS

5.11.4.1 Delvs, requête SEDRIC

Test de validité du message client.

Suppression de l'écran virtuel de nom 'VSNAME' dans le tableau des écrans virtuels à l'emplacement 'VSNUM'. On le remplace par des zéros.

Transfert du message à la tâche GEV1.

5.11.4.2 Delvs, tâche GEV1

Purge de la file d'attente pour le contexte de numéro 'VSNUM'.

5.11.4.3 Delvs, tâche GES1

Réagencement de l'écran.

GES1 dispose d'une liste par numéro de lignes. Elle comprend les différents segments par ordre de recouvrement. Celui en tête de liste contient la ligne visible. S'il est supprimé, la ligne appartenant à son successeur est réaffichée.
Ces lignes sont mémorisées dans les écrans virtuels associés aux segments.
GES1 parcourt toutes les listes et demande à GEV2 le réaffichage des lignes concernées.
Cela se traduit par l'envoi de messages 'DISP' internes qui ont la structure suivante :

Code message DISP

Numéro du segment

Numéro de la ligne par rapport au début du segment.

La tâche GEV2 acquiert dans l'écran virtuel le contenu de la ligne concernée. Elle forme, avec, un message 'REDISP' qu'elle transmet à GEV1.

Il est alors traité comme une primitive 'DISP' sans la recopie du texte dans l'écran virtuel.

5.11.4.4 Delvs, tâches PAV1 et TIO1

Elles traitent les messages d'affichage transmis par GES1 comme pour une primitive 'DISP'.

5.12 REALISATION DE L'INTERFACE USAGER

Cela correspond aux différents modules déroulés pour prendre en compte les décisions de l'opérateur transmises par les touches fonctions que nous avons présentées au chapitre 4.

5.12.1 Touche SEND

5.12.1.1 Touche SEND tâche TI02

Cette touche provoque l'envoi de tous les caractères non protégés du segment ainsi que les délimiteurs de lignes et de zones.
Ils sont rangés dans le buffer dont l'adresse a été fournie par TI01 dans l'IOCB.

La fin d'échange provoque le réveil de la tâche TI02 en attente sur le sémaphore spécifié dans le mot 6 de l'IOCB.

Le message transmis par la tâche TI02 à la tâche PAV2 comportent les éléments suivants :

- type de l'échange terminé (entrée ou sortie)
- compte-rendu d'échange, soit :
  - nombre de caractères transmis, ou,
  - nature du défaut.

5.12.1.2 Touche SEND, tâche PAV2

Ici, on dépouille le buffer pour reconstruire les zones intéressant le module client.

En effet, il est structuré ainsi :

‹identification de la touche›

(‹délimiteurs de fin de ligne›)*

‹délimiteur de début de zone›

‹texte de la zone›)*

‹caractère fin de message›

PAV2 construit le message suivant :

‹primitive en cours de la fin d'échange›

‹compte-rendu›

‹numéro de touche›

(‹numéro de ligne de la première zone

‹numéro de colonne›

‹dernière zone ou non› (booléen)

‹longueur de la zone›

‹texte›)*

‹code fin de message›

Et le transmet à la tâche GES2.

Remarques :
Ce que nous venons de décrire pour les tâches TI02 et PAV2 est vrai pour toutes les touches fonction exceptée celle provoquant le changement de segment. La suite de l'exposé en sera allégée.

5.12.1.3 Touche SEND, tâche GES2

Récupération du numéro de segment par déchaînage dans la liste affectée au contexte traité.

Insertion de ce numéro dans le message reçu.

Transfert de celui-ci à la tâche GEV2.

5.12.1.4 Touche SEND, tâche GEV2

Mise à jour de l'écran virtuel à l'aide du message reçu.

Le traitement est en tout point semblable à celui effectué pour la primitive 'DISP'.

Le numéro de segment est remplacé dans le message par le numéro d'écran virtuel.

Le numéro de touche est lui remplacé par le numéro de la primitive en cours sur l'écran virtuel auquel on ajoute 100 pour indiquer qu'on répond à cette primitive.

Appel de SEDRIC en fournissant, en paramètre, le message que l'on vient de construire.

5.12.1.5 Touche SEND, requête SEDRIC

Adressage du contexte en fonction du numéro d'écran virtuel contenu dans le message.

Si le message est une réponse à un menu alors

Transfert à la tâche ALFEGA

Sinon (réponse à RECEIVE ou CREATVS)

Ajoût du nom de l'écran virtuel au message reçu,

Transfert de celui-ci dans le buffer spécifié à l'appel

Réveil du module client.

5.12.1.6 Touche SEND, tâche ALFEGA

Dans ce cas la touche a servi à faire connaître la réponse à un 'menu'.

Avant de décrire le traitement effectué nous allons présenter une structure de données utilisée.

Elle appartient au menu 'historique', c'est la table des numéros d'items. Elle contient tous les numéro d'items choisis lors du parcours de l'arbre. Ils sont classés par ordre de profondeur.

La tâche ALFEGA utilise cette table pour reconstituer le parcours théorique après une demande d'historique et le départ d'un nouveau parcours dans l'arbre.

Voici comment est implémentée cette table dans l'historique :

Mot 8 du menu : adresse de la table
(index dans l'article pointant théoriquement derrière le texte affiché initialement pour la variable réponse (blancs).

Mot 0 de la table : numéro de l'item choisi dans le menu racine

Mot 1 de la table : numéro de l'item choisi dans le menu fils.
Etc...

Exemple :

Soit le parcours suivant :

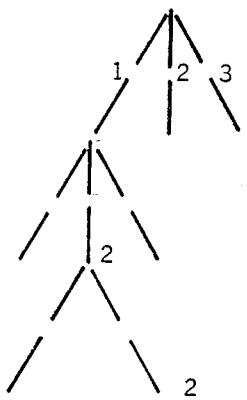


Fig. 32

La table contient :

Mot 0 : 1
Mot 1 : 2
Mot 2 : 2

Revenons à la tâche ALFEGA. Voici l'algorithme déroulé :

Acquisition de la réponse à partir du message reçu.

Mise à jour de la table des numéros d'items

Incrémentation de la profondeur dans l'arbre des menus.

Ajoût du numéro d'item dans le message réponse.

Création de l'historique s'il n'existe pas.

Ajoût dans l'historique de l'item choisi, soit :

Incrémentation du nombre d'items

Libellé : libellé du choix concaténé au nom du menu courant

Table-de-décision (prof) := nom du menu courant.

Lecture de la table de décision

Si ce n'est pas un choix terminal

Traitement du menu ou de la grille de paramètre tel que nous l'avons décrit dans la primitive 'CREATVS'.

Sinon

Appel de SEDRIC avec comme paramètre le message relatant les choix.

Le mot zéro du message contient le code de 'CREATVS' plus 100 indiquant un retour.

Remarque :
L'article historique a pour nom 'HISTORXY' où XY est le numéro de l'écran virtuel associé.

5.12.2 Touche 'MENU-HISTORIQUE

Si l'on travaille par menu aucune donnée n'est transmise sinon le contenu du segment est transmis de la même manière que pour la touche 'SEND'.
Bien évidemment le numéro de la touche est toujours transmis.

5.12.2.1 touche 'MENU-HISTORIQUE, tâche TIO2 et PAV2

Le traitement est identique à celui effectué pour la touche 'SEND'.

5.12.2.2 Touche MENU-HISTORIQUE, tâche GES2

Déchaînage du numéro de segment

Transmission du message à la tâche GEV2

5.12.2.3 Touche MENU-HISTORIQUE, tâche GEV2

Remplacement du numéro de segment par le numéro d'écran virtuel dans le message reçu

Mise à jour de l'écran virtuel

Transmission du message à la tâche 'ALFEGA'.

5.12.2.4 Touche MENU-HISTORIQUE, tâche 'ALFEGA'

Le traitement se fait en plusieurs étapes.

1. Affichage du menu historique et demande d'acquisition de la réponse (primitive interne 'RCVMENU').

2. En retour :

Reconstitution du message théorique à transmettre à partir de la table des numéros d'items et de la réponse fournie. Elle indique la profondeur à partir de laquelle on veut repartir dans l'arbre des menus.

3. Traitement du menu demandé. Son nom est donné par le numéro d'item choisi et par la table de décision de l'historique dans laquelle il indique l'entrée (index).

5.12.3 Sélection de segment

Il faut commencer par traiter le break.

La tâche spécifique signale à TIO2 l'arrivée du break.

Cette dernière transmet à la tâche PAV2 le message suivant :

- mot 0 : nature de l'échange interrompu (entrée ou sortie)

- mot 1 : compte-rendu = défaut de type break.

Cette dernière tâche demande à PAV1 d'envoyer vers la console les codes provoquant :

1. Le déverrouillage de toutes les lignes de l'écran pour permettre à l'usager de positionner le curseur sur le segment désiré.

2. Une entrée

L'opérateur peut répondre à cette sollicitation de deux manières différentes

1. Il positionne le curseur sur le segment désiré et frappe sur la touche 'SELSEG'.

2. Il demande la liste des segments (touche MENUSEL).

Etudions maintenant chacune des alternatives.

5.12.3.1 Touche 'SELSEG'

Elle est programmée de telle sorte que sont envoyés à la tâche PAV2 les codes suivants :

- numéro de la touche

- numéro de ligne et

- numéro de colonne du curseur.

Ces codes sont transmis par message à la tâche GES2.

Rappelons que celle-ci dispose de deux tables indiquant pour chaque segment

- le numéro de la première et de la dernière ligne.

- le numéro de la première et de la dernière colonne.

Disposant dans le message reçu des coordonnées de sélection, GES2 peut donc retrouver le numéro du segment désiré.

Le message transmis à GEV2 contient :

- le numéro de la fonction 'SELSEG'

- le numéro du segment sélectionné.

Celle-ci l'interprète comme étant une demande d'affichage de la fenêtre logique correspondante.
Elle envoie, à la tâche GEV1, les primitives 'REDISP' nécessaires. Elles sont alors traitées en tant que telles par GEV1, GES1, etc... (CF DELVS).

Toutefois, si une primitive 'RECEIVE' ou bien RCVMENU, interne à SEDRIC, était en cours d'exécution sur la fenêtre réaffichée lorsqu'elle avait été quittée, cette primitive est redemandée via GEV1.

5.12.3.2 Touche 'MENUSEL'

Elle est programmée de telle sorte que n'est envoyé à la tâche PAV2 que le code identifiant la touche.
Cette tâche le transmet à GES2.
Cette dernière envoie les messages d'affichage élaborant le menu formé des noms des différents segments.
Elle entre alors en état de sélection pour la console concernée.
Cela se traduit notamment par l'envoi vers les couches en aval d'un message de réception tel que celui envoyé lors d'une primitive 'RECEIVE'. Il est traité de la même manière. A la réception de la réponse, la tâche GES1 envoie à GEV2 le message de sélection décrit plus haut.
On se retrouve alors dans la première alternative.

5.12.4 Déplacement de la fenêtre logique

Deux touches sont mises à la disposition de l'opérateur. Ce sont :

WINDOW UP (WU) qui permet de déplacer la fenêtre logique vers le haut dans l'écran virtuel.

WINDOW DOWN (WD) qui assure un déplacement vers le bas.

En voici l'implémentation :

5.12.4.1 Touches WU et WD, tâche TIO2 et PAV2

Elles traitent ces deux touches d'une manière semblable à la touche 'SEND. Dans les messages transmis, seul le code identifiant le type de la touche change.

5.12.4.2 Touches WU et WD, tâche GES2

Elle reçoit de la tâche PAV2 le message suivant :

- code de la fonction
- contenu du segment de façon identique à celui transmis pour la touche SEND.

Elle déchaine le segment (il n'est plus en attente d'entrée).

Elle insère son numéro dans le message reçu,

Transmet le message ainsi modifié à la tâche GEV2.

5.12.4.3 Touches WU et WD, tâches GEV1 et GEV2

Le traitement se fait en deux étapes :

1. Mise à jour de l'écran virtuel. Le traitement est identique à celui effectué pour la touche 'SEND'.
2. Traitement spécifique de la touche WU ou WD

Pour cela :

Définition d'une nouvelle fenêtre logique par modification de ses pointeurs de première et de dernière ligne. Ils sont incrémentés ou décrémentés du quantum (1/3 de la taille) suivant que la touche traitée est WU ou WD. Cette modification est toujours possible car l'écran virtuel est géré en mode rondeau.

Si la touche traitée est WD :

- les lignes appartenant au quantum de translation sont recopiées dans le fichier de l'écran virtuel. Ces lignes ne seront plus visibles (CF présentation de WD (paragraphe 4.3).

Dans tous les cas une nouvelle fenêtre est lue à partir du même fichier. Elle vient écraser l'ancienne fenêtre logique.

Construction de messages internes 'REDISP', transmis à GEV1 pour afficher la nouvelle fenêtre.
Comme nous l'avons déjà noté lors de la primitive DELVS, ces messages sont semblables à ceux transmis pour spécifier une primitive 'DISP'. Seul le code change. La différence au niveau du traitement réside dans la non recopie du texte du message dans l'écran virtuel.

6 EXTENSIONS

La version actuelle de SEDRIC est une maquette et non un produit au sens industriel du terme. De de fait, il nous a semblé bon de réserver quelques lignes aux améliorations que nous eussions aimé faire, en vue d'une exploitation plus commerciale.

6.1 LES MENUS

Actuellement, l'indication du choix se fait en frappant sur le clavier le numéro de l'item correspondant.

Le désigner à l'aide du curseur sur notre matériel, d'une souris, d'un 'light pen' ou du doigt sur d'autre matériel serait plus naturel.

C'est d'ailleurs ainsi que l'on sélectionne un segment. Il est aisé de le généraliser aux menus. D'autant plus que le numéro de l'item ne fait pas partie du texte, il est calculé et rajouté par SEDRIC. Seule la conviction première que l'implémentation actuelle était plus conforme à l'habitude informatique et plus pratique avec le matériel utilisé nous a conduit à la solution adoptée.

6.2 LA CONSOLE

Nous avons vu que s'il est possible de dessiner sur l'écran d'une console télévidéo des fenêtres rectangulaires de taille quelconque, il s'avère difficile d'assurer un dialogue multifenêtres et multiactivités ayant des performances acceptables. C'est pourquoi les fenêtres de SEDRIC sont, sur la réalisation faite, une suite de lignes contigues. La puissance du dialogue en est altérée. C'est pourquoi, il nous aurait été utile de disposer de console et de logiciel graphiques.
Une télévidéo graphique et la bibliothèque FGL de SEMS seraient suffisantes. (FGL : Fortran Graphic Library).

6.3 LA FENETRE COMMANDE

Un module client peut demander à SEDRIC de lui créer une fenêtre réservée à un dialogue par commande. Un contrôle syntaxique pourrait être fait par SEDRIC.

Il faudrait pour cela que le module client fournisse la grammaire du langage de commande en paramètre de la commande CREATVS. Ce pourrait être le nom d'un fichier ou l'article d'un fichier contenant toutes les grammaires codées sous forme de tables ou de langages intermédiaires.

Comme le sont les menus, ces grammaires seraient spécifiées au préalable. Cela se ferait à l'aide d'un langage de description imposé par SEDRIC.

Une tâche spécifique, véritable interprète de grammaires, traduirait chaque règle ou élément de règle, pour former la grammaire du langage de commande.

A la fenêtre commande, serait alors attachée une tâche particulière comme l'est ALFEGA pour les menus (ou bien un module 'd'alfega').

6.4 TOUCHE HELP

Nous avons noté dans le premier chapitre combien il était appréciable pour l'opérateur de disposer d'une aide dite "on line'. Dans une fenêtre commande SEDRIC lui-même pourrait assurer totalement cette fonctionnalité.

Elle peut prendre diverses formes :

- affichage de la syntaxe à la suite d'une erreur de l'opérateur et sur sa demande.
- affichage de la syntaxe d'une commande spécifiée par l'opérateur.
- etc.

Pour un menu la touche 'Help' n'est pas d'une grande utilité. L'opérateur est déjà guidé.

Pour d'autres applications permises par SEDRIC telle que la saisie de grilles la gestion du 'help' ne peut être que laissée à la charge du module client. Il demandera alors à SEDRIC de valider une touche fonction et il l'interprètera comme une demande d'aide.

6.5 LE PASSAGE DE PARAMETRES

Pour garder aux paramètres leur caractère facultatif, permis par l'interface message tout en agrémentant la programmation des modules clients il serait intéressant de réaliser un pré-processeur du langage interface. Il permettrait de fournir les paramètres à l'aide de mots-clé.

Ainsi la primitive de création de l'écran virtuel aurait la forme suivante.

CREATVS VSNAME = (,VSSIZE=)(,SEGSIZE=)(,MENUNAME=)(,DELIM ,,,)(,CMOMODE), DEVICENUM=.
Sont entre parenthéses les paramètres facultatifs.

7 CONCLUSION

Preuve de faisabilité une maquette se doit, également, d'être source d'applications.

Le caractère multiactivité et multiconsole de SEDRIC nous a conduit à faire une étude pour qu'il devienne un serveur intégré dans le cadre d'un système temps partagé.
Nous avons montré, au début de notre exposé que SEDRIC nécessite le support d'un système multi-tâche.
L'objet de l'étude a été de proposer les modifications à apporter au système. SEDRIC en devien t un serveur . L'ancienne tâche dialogue prend le statut de module client.

Consultations, éditions et compilations simultanées de plusieurs programmes. Visualisation d'informations systèmes pendant l'exécution d'un processeur et plus généralement l'agrément du dialogue dû aux fenêtres multiples, au multi-activité et à l'ensemble des fonctionnalités de SEDRIC sont séduisants pour le programmeur. Ils pourraient faire de SEDRIC un outil de base dans l'élaboration d'un poste de programmeur.
Si dans ce but, l'intégration de SEDRIC sur une nouvelle machine peut s'avérer coûteuse compte-tenu des retombées commerciales limitées, elle peut paraître plus intéressante sur une machine nouvelle pas encore totalement figée.

SEDRIC est un serveur de dialogue. Sa vocation est d'être suffisamment universel pour interfacer un large ensemble d'applications. Il facilite leur conception en les déchargeant de la gestion du dialogue dont il contribue à l'uniformité, critère de qualité. Il est, par ailleurs, très adapté à la gestion de grille.

Enfin, SEDRIC s'étant inspiré de Smalltalk, avec une portée certes limitée, nous aimerions le considérer comme un point de départ, modeste, vers l'élaboration d'un poste de travail tels que le sont Perq, Star, Apollo et autres Lisa.

8 ANNEXES

8.1 STRUCTURE D'UNE TACHE SEDRIC

Un module sur le SOLAR dispose de quatre sections de données.
Ce sont :

1) La section des tables accessibles par des pointeurs situés dans les autres sections.

2) La section des données communes pointée par le registre de base nommé : 'registre C'.
Elle contient généralement les données globales.

3) La section des données locales pointée par le registre 'L'.
Chaque sous-programme peut avoir une section de données locales.

4) La section contenant des données de travail.
Elle est pointée par le registre nommé : 'registre W'.

La taille des sections basées est limitée à 256 mots (de 16 bits).

Il faut toutefois noter que la présentation que nous venons de faire est théorique.
Aucun contrôle n'est fait par la machine sur la nature des données contenues dans chaque section (le pourrait-elle ?).
Dès lors, rien n'empêche l'usager de mettre des données communes dans la section locale et réciproquement.

Remarque :

Les trois sections s'appellent encore :

- common section (données communes)

- local section (données locales)

- working section (données de travail).

Application à SEDRIC

Chaque tâche ou la requête SEDRIC a la même structure.
Soit :

Une section table contenant les tableaux statiques.

Une 'common section' contenant généralement les données communes à la couche contenant la tâche.

Eventuellement, une 'working section', dans le cas des tâches PAV1 et PAV2 quicontient les tables servant à assurer la fonction protocole d'appareil virtuel. Il y a une 'working section' par type de console.

Une 'local section' qui contient les contextes d'exécutions.
Elle est située en fond de partition. Une 'dummy section' est plaquée dessus.
L'initialisation de la base L, adressant cette section, est faite

par la tâche elle-même à l'aide du numéro de contexte et d'un tableau d'adressage.

Voici la structure des tâches PAV1 et PAV2.

ORGANISATION DE LA PARTITION PAV

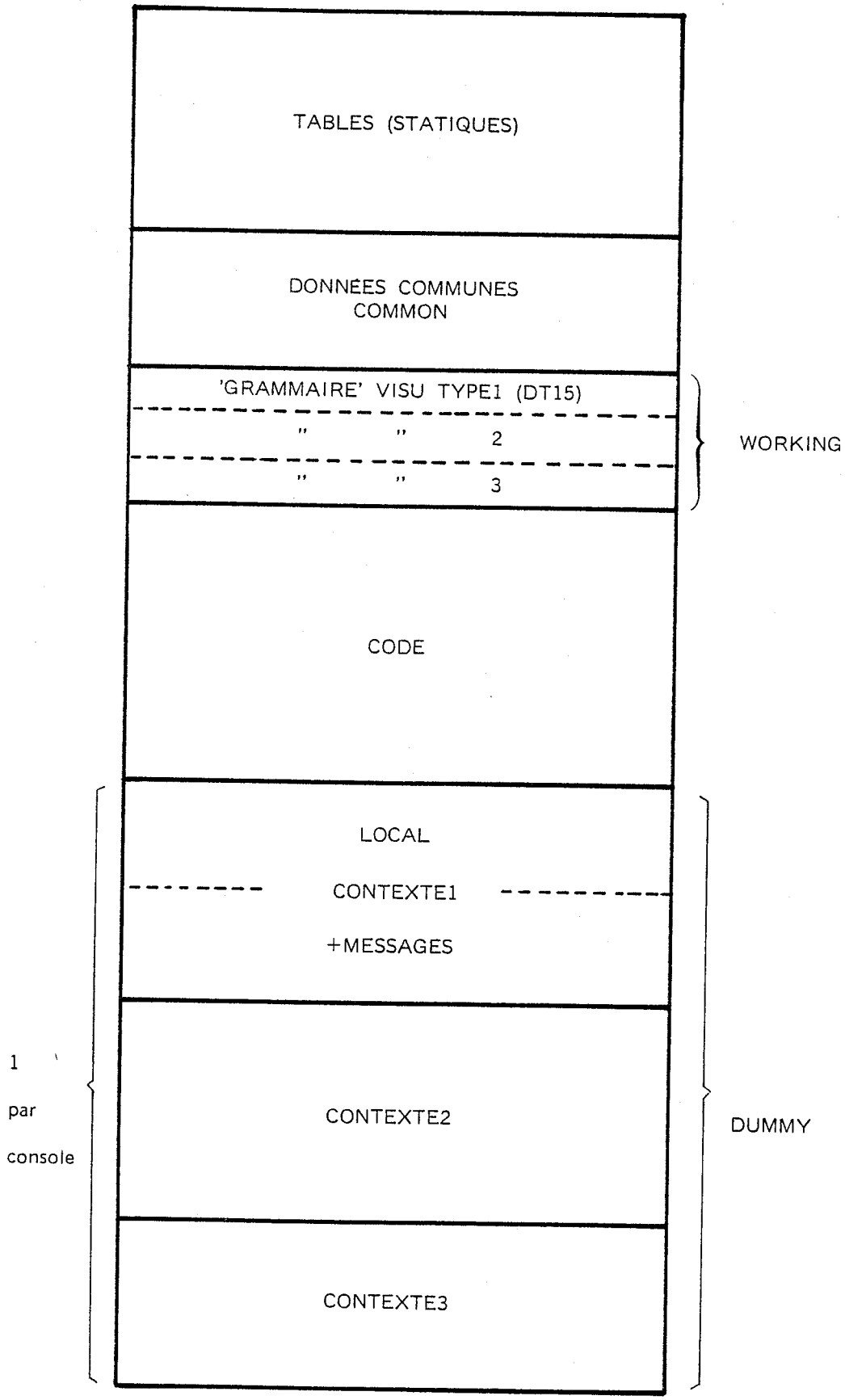


Fig. 33

Voici, en prenant l'exemple de la requête SEDRIC, comment sont adressés les contextes.

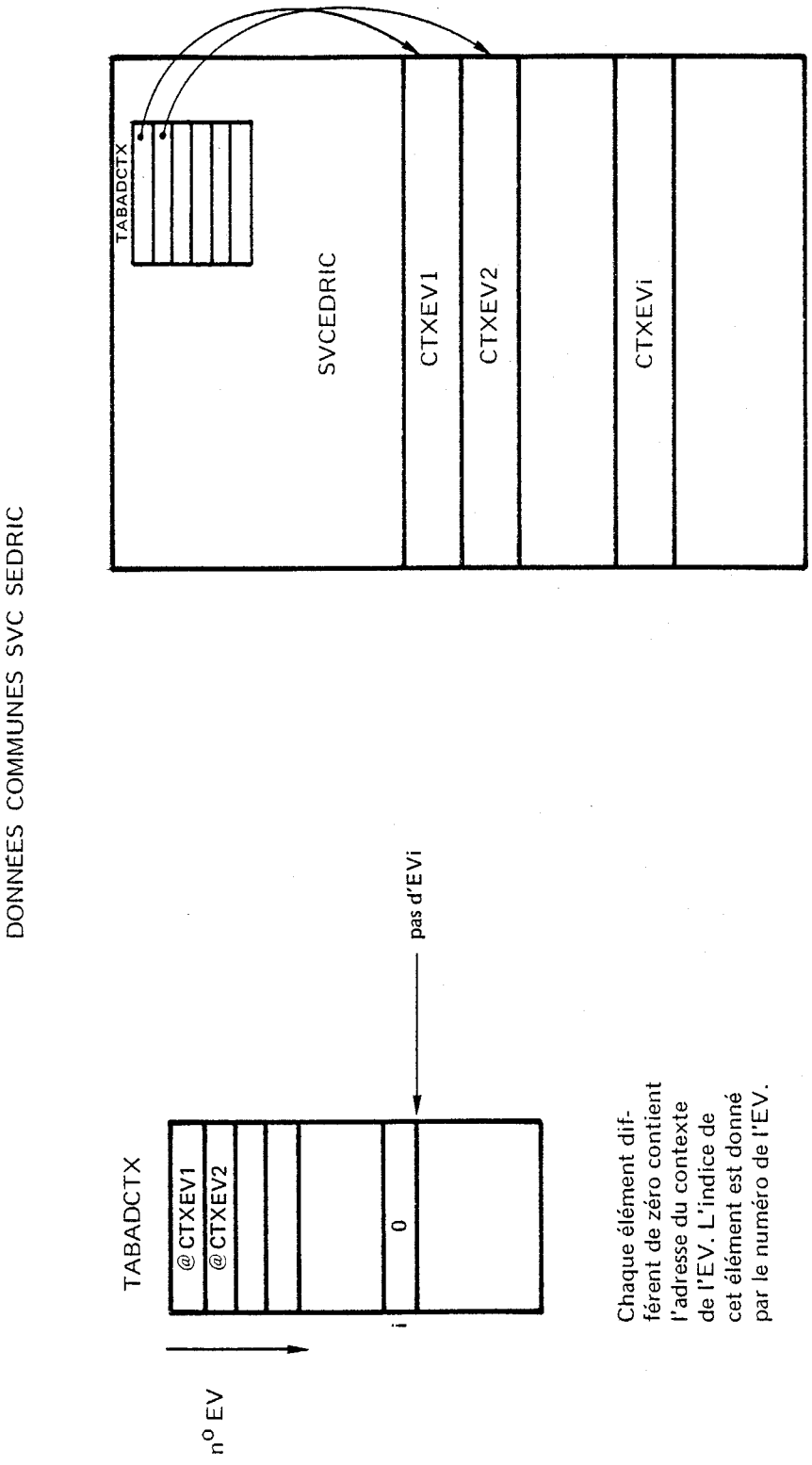


Fig. 34

8.2 INTEGRATION DE SEDRIC SOUS LES SYSTEMES TEMPS PARTAGE : TSM OU TSF

Définissons auparavant quelques termes spécifiques du vocabulaire 'SOLAR'.

- FU : Unité Fonctionnelle, il est pris ici en tant que numéro de console.
- tâche TSM FINECH : tâche traitant l'évènement fin d'échange dans TSM.
- FU TSM : numéro de console pour TSM, appelé aussi FU symbolique.
- FU console : numéro de console réel.

Rappelons également un des principes de fonctionnement des systèmes TSM/TSF.

C'est la bijection établie entre l'usager système (l'activité au sens SEDRIC) et la FU console.

Cette relation est établie à l'entrée dans le système (login). Elle est rompue au moment de la déconnexion ('logout').

On conçoit, dès lors, la difficulté à établir un dialogue dit multiactivité.

La proposition que nous faisons est destinée à contourner cette difficulté.

Les modules clients accèdent à SEDRIC par des primitives relatives à un écran virtuel. Lorsque ces requêtes nécessitent un échange avec la visu, le serveur les transforme en une requête à IOCS dans laquelle la FU est une FU TSM "symbolique".

Un module, appelé SEDRIOCS, piège la requête afin de remplacer la FU TSM par la FU console. Inversement en fin d'échange SEDRIOCS modifie la FU pour que TSM réactive le bon module client. (Il doit donc intervenir avant la tâche TSM FINECH) SEDRIOCS ne doit pas être un chapeau. Il doit intervenir après TSM afin que, pour celui-ci tout soit transparent.

Nous venons de voir que pour découper un écran en plusieurs segments il fallait qu'à chaque segment créé corresponde une FU.

Figeant, pour la requête, le nombre des segments activables simultanément nous décrivons, à la génération, plusieurs FU pour chaque console. Il y aura une FU dite mère (FU IOCS) et n-1 FU dite fille (FU TSM symbolique).

Nous allons maintenant exposer le fonctionnement du système pour une console.

Dans un premier temps, les échanges se font sans solliciter SEDRIC. C'est le dialogue système.

Dès le premier appel, SEDRIC prend comme FU segment la FU courante. Ainsi le premier utilisateur SEDRIC est logiquement pris en compte par le système (TSF, TSM).

SEDRIC crée implicitement un segment système que SEDRIOCS associe à la première FU-fille libre, ici S2.

Il y a donc un segment explicitement associé à un module client, et, un autre réservé implicitement au système.

Quand l'opérateur demande à changer de segment pour activer un autre module client on le commute sur le segment système à qui on a associé la FU S2. Pour cela on simule le 'break' et le 'login' sur la FU S2. C'est à cette FU que sera associé le module client lancé par la suite.

De nouveau, on crée implicitement un segment système associé à la FU S3.

Il y a alors trois segments :

S1 <---> MC1

S2 <---> MC2

S3 <---> système

Lorsque le module client actif rend la main à TSF/TSM celui-ci passe en entrée de commandes sur le segment.

Si le deuxième module client, MC2, termine son travail le dialogue système se met en entrée de commande sur S2.

Pour éviter d'avoir plusieurs dialogues sur le même écran, et cela, indépendamment de SEDRIC, il faut que SEDRIOCS réponde par 'logo' à la première entrée qui suit la primitive DEL-EV.

Remarque :
La simulation du 'logo' est déjà faite dans TSM/TSF.
Il en est de même pour le 'break' et le 'login' (appelé START).
On pourrait donc utiliser ce qui est déjà fait.
Cela pourrait se faire sous forme de deux requêtes TSM-TSF.
L'une pour le 'login' et le 'break', l'autre pour le 'logo'.

Fonctionnement de SEDRIC vis-à-vis de TSM

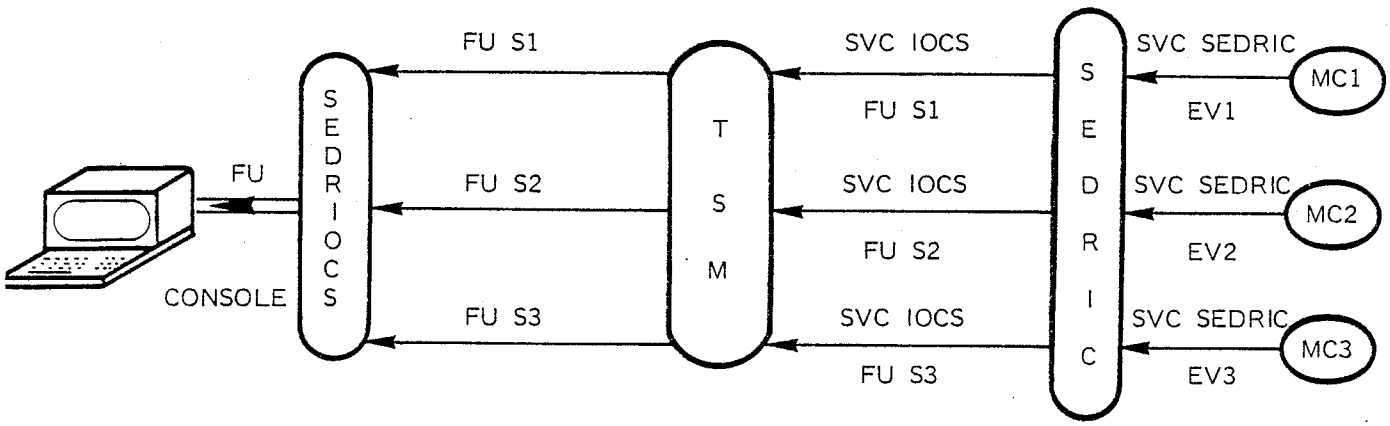


Fig. 35

FU Si : FU TSM ou FU symbolique de numéro i.
EVi : écran virtuel de numéro i.
MCi : module client de numéro i.

8.3 STRUCTURE D'ACCUEIL

8.3.1 Définition

Nous appelons structure d'accueil une structure de données destinée à recevoir des informations provenant d'une source extérieure, ici la console.

De par la structure du message d'appel de la primitive 'DISP' et du message retourné sur la primitive 'RECEIVE', SEDRIC est particulièrement adapté à la saisie de données formattées (grille de saisie).

Une grille est un ensemble structuré de zones de saisie.

Partant de ce principe, on conçoit que l'on puisse faire correspondre à une grille de saisie une structure, telle, celles implémentées dans PASCAL.

Considérons la grille suivante :

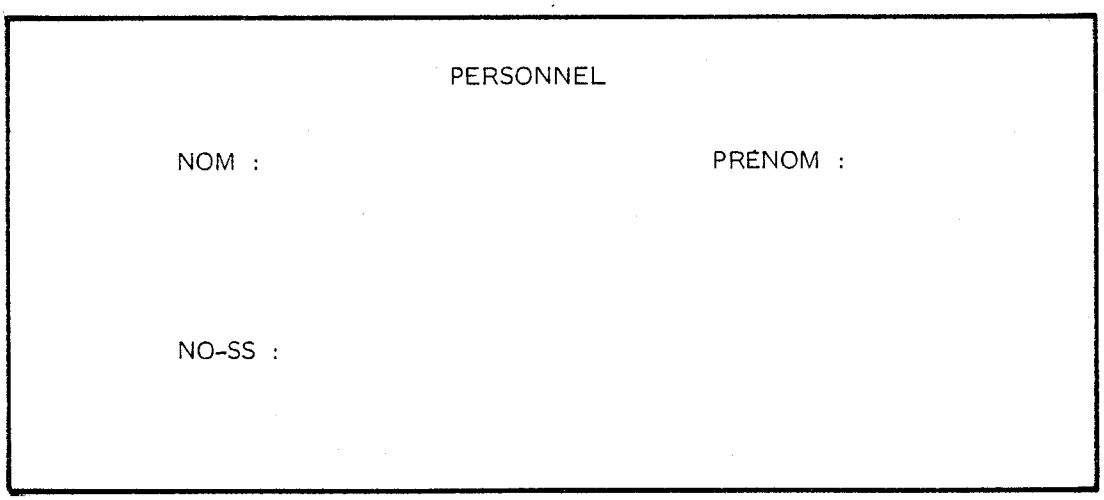


Fig. 36

La structure correspondante serait en PASCAL :

```
TYPE GRILLE = RECORD
                    NOM : ALFA;
                    PRENOM : ALFA;
                    NO-SS : ALFA;
          END;
VAR PERSONNEL : GRILLE;
```

Les champs sont tous de type 'ALPHA' car les zones sont considérées à ce niveau comme une suite de caractères.

Pour autoriser le processus d'association des grilles et des structures nous avons développé un module chargé de fabriquer les grilles de saisie.

Il est fortement inspiré du module de fabrication des menus que nous avons déjà décrit.

Connaissant les grilles conçues par l'usager, il engendre le texte décrivant

la structure adéquate ainsi que les instructions nécessaires pour acquérir les données à partir des messages fournis par SEDRIC .

Contraintes :

Chaque grille doit avoir un nom. Il devient le nom de la stucture d'accueil.

Chaque zone de saisie doit avoir un libellé on s'en sert pour nommer les champs dans la structure.

9 BIBLIOGRAPHIE

SMALLTALK, BYTE AOUT 81.

SOFTWARE-PRACTICE AND EXPERIENCE
VOLUME 12 NO 3 MARS 82
DESCRIPTION OF A MENU CREATION AND INTERPRETATION SYSTEM.
MICHAEL J. HEFFLER, BELL LABORATORIES.

COMMUNICATION OF THE ACM
VOLUME 25 NO 7 JUILLET 82
CONTROLLING THE COMPLEXITY OF MENU NETWORK.

ADELE : RAPPORT DE RECHERCHE.
M. HERRMANN, J. RAYMOND, IMAG.

PACTOLE, E. D. F.

KAYAK
BULLETIN DE LIAISON DE L'INRIA (NO 69 ET 70)

CALPIN
DOCUMENT DE REFLEXION COLLECTIVE SUR LES CRITERES DE QUALITE DU DIALOGUE HOMME-MACHINE DANS UN SYSTEME INFORMATIQUE.
SEMS.

Manuel de référence langage PL16
Documentation SEMS

Manuel de référence PASCAL
Documentation SEMS

Manuel de référence RTES/D
Documentation SEMS

Manuel d'utilisation RTES/D
Documentation SEMS

Manuel de référence IOCS
Documentation SEMS

Manuel d'utilisation IOCS
Documentation SEMS

Handbook SOLAR
Documentation SEMS.