UNIVERSITE DE NANCY I

U.E.R. SCIENCES MATHEMATIQUES

LE SYSTEME SAFRAN

THESE

pour l'obtention du

DIPLÔME DE DOCTEUR INGENIEUR

Soutenue le 23 Juin 1978

par

Michel DUBUIT

Ingénieur E.N.S.M.I.M.

MEMBRES DU JURY :

Président : M. C. PAIR

Examinateurs : MM. J.C. DERNIAME

M. GRIFFITHS

J.M. VILLARD

UNIVERSITE DE NANCY I

U.E.R SCIENCES MATHEMATIQUES

LE SYSTEME SAFRAN

THESE

pour l'obtention du

DIPLÔME DE DOCTEUR INGENIEUR

Soutenue le 23 Juin 1978

par

Michel DUBUIT

Ingénieur E.N.S.M.I.M.

MEMBRES DU JURY :

Président : M. C. PAIR

Examinateurs : MM. J.C. DERNIAME
M. GRIFFITHS
J.M. VILLARD

Je remercie vivement Monsieur le Professeur PAIR qui, après avoir été mon patron à l'Institut Universitaire de Calcul Automatique, m'a fait l'honneur de présider le jury.

Ce travail a été effectué sous le contrôle de Monsieur DENNIARD. Je tiens à lui exprimer ma plus sincère gratitude pour la gentillesse avec laquelle il m'a prodigué aide et conseils pendant toute la durée de ma recherche.

Monsieur GRIFFITHS, directeur de l'Institut Universitaire de Calcul Automatique, m'a montré une grande compréhension lors de la réalisation de cette thèse. Il a bien voulu juger mon travail. Qu'il trouve ici l'expression de ma reconnaissance.

Grand merci à Monsieur VILLARD, qui a volontiers accepté d'effectuer le voyage de Rennes à Nancy pour me faire l'honneur de participer au jury.

Enfin, je tiens à remercier Mesdames LAMBERT, MARCHAND, DECOISY, qui ont pris en charge la réalisation materielle de cette thèse.

SOMMAIRE

HISTORIQUE

-Juin 1971 :

Pour faciliter l'enseignement de la programmation élémentaire, Madame DENDIEN propose un ensemble de programmes permettant l'interprétation de langages-machines de complexité croissante : les Machines Simulées. Ces programmes sont adaptés au mode de traitement en continu offert par les systèmes B.P.M., puis SIRIS 7 de l'ordinateur CII IO070, disponibles à l' I. U. C. A. de NANCY.

-Octobre 1972 :

Monsieur DENNIAME suggère le développement d'un niveau supplémentaire de code permettant l'étude pratique des entrées-sorties, et des mécanismes spéciaux (canaux, interruptions, déroutements, commutation d'état). Le programme est au point en Juin 1973 grâce aux conseils de Madame DENDIEN, que je tiens à remercier ici pour l'aide qu'elle m'a accordée.

-Octobre 1973 :

Monsieur DENNIAME demande la réalisation d'une version conversationnelle des Machines Simulées, offrant des possibilités de mise au point du type de celles offertes par les panneaux de commande des ordinateurs (ce type de fonctions a déjà été offert dans les compilateurs incrémentaux).
Pour qu'une première version de ces programmes soit rapidement opérationnelle, le système général TEMPS PARTAGE SIRIS 7 de l' I. U. C. A. de NANCY sera utilisé.

-Avril 1974 :

Un contrat est passé avec SFER pour cette réalisation. Il permet à Monsieur MERCIER de participer à la mise au point de ces programmes.

-Septembre 1975 :

Les programmes conversationnels sont disponibles.
Je tiens à exprimer ici toute ma reconnaissance à Monsieur MERCIER pour sa collaboration efficace : Durant toute la durée de cette réalisation, il a toujours accepté de participer aux travaux necessaires, même aux plus ingrats. Son travail tient une large place dans la réussite de ce projet.

-Octobre I975 :

Les Machines Simulées conversationnelles sont utilisées pour la première fois au cours des séances de travaux pratiques de la maîtrise d'informatique.
Une séance de travaux pratiques utilisant les machines simulées est un cas très particulier d'exploitation conversationnelle.
Le système TEMPS PARTAGE, d'utilisation générale, permet cette exploitation, mais ne semble pas dans ce cas garantir une utilisation optimale de l'ordinateur.
Les caractéristiques particulières de ce type d'exploitation sont alors dégagées (Chapitre I) . Elles sont communes avec d'autres applications : édition de textes, ...
Il semble interessant d'envisager l'utilisation d'un système plus spécialisé, mettant en oeuvre, entre autres, la réentrance des programmes.
Parmi les produits CII standard, aucun ne présente les caractéristiques requises (Chapitre II). Il ne semble pas, non plus, qu'un système de ce type ait déjà été proposé à titre expérimental : Ce système est intermédiaire entre les systèmes généraux Temps Partagé et les systèmes très spécialisés de Gestion De Transactions.
Il semble donc utile de développer un logiciel de ce type (Chapitre III). Il éviterait à tous les usagers potentiels de programmer des fonctions équivalentes dans leurs applications
Une première version du système SAFRAN est ainsi mise à l'étude.

-Juin I977 :

La première version du système est opérationnelle sur l' IRIS 80 de l' I. U. C. A. de NANCY. Sa mise au point a parfois causé quelques désordres en salle machine ... L'équipe d'exploitation les a toujours considérés avec philosophie, et a participé de bonne grâce à mes recherches. Qu'elle trouve ici l'expression de mes remerciements sincères.

-Janvier I978 :

Les Machines Simulées sont modifiées pour être exploitées sous le contrôle du moniteur SAFRAN.

-Courant I978 :

Une deuxième version du système sera réalisée, utilisant les services offerts par la version CIO du système SIRIS 8.

Elle permettra une exploitation plus souple et une ouverture à un large éventail de correspondants. (Réseau CYCLADES)
Cette version devrait être mise en exploitation réelle en séances de travaux pratiques en Octobre I978.

Sur beaucoup de petits systèmes actuels, comme jadis sur les premiers ordinateurs, les usagers contrôlent directement l'exécution de leurs programmes. Les entrées de données, les sorties de résultats s'effectuant au rythme du calcul, l'évolution du programme peut être suivie en permanence, voire orientée en des directions différentes en fonction des résultats déjà obtenus.
Afin d'offrir des possibilités de dialogue semblables, et ce à plusieurs usagers simultanément, les gros systèmes actuels contiennent généralement un ou plusieurs services permettant l'exécution d'applications conversationnelles.
Le système SIRIS 8 standard offre plusieurs services de ce type :

- le sous-système général "temps partagé"
- des sous-systèmes spécialisés (gestion de transactions, gestion de banques de données)

Pour la mise en oeuvre d'applications conversationnelles partagées n'entrant pas dans les catégories transactions ou interrogation de banque de données, aucun sous-système spécialisé n'est disponible : seul le "temps partagé" peut être facilement utilisé.
Or ce service, d'utilisation générale, ne tient pas compte des particularités d'exploitation des applications qu'il contrôle. Les algorithmes utilisés ne sont donc pas optimaux pour toutes les applications. Le sous-système SAFRAN présenté ici (Système pour des Applications Fermées sur Réseau d'Accès iNtéractifs) est un moniteur spécialisé, qui concerne les applications conversationnelles présentant les caractéristiques suivantes :

- plusieurs usagers demandent simultanément les services de l'application.
- un ensemble limité de services prédéfinis est offert aux usagers.
- la logique des échanges usagers $\Longleftrightarrow$ application n'est pas régie par des règles simples.

En tenant compte des caractéristiques très particulières de ce type d'applications, SAFRAN tente d'en améliorer l'exploitation en jouant sur deux tableaux :

- diminution des ressources nécessaires, donc du coût de mise en oeuvre.
- amélioration de la qualité du service offert aux usager (temps de réponse).

Les chapitres suivants mettent plus précisément en évidence son utilité et décrivent ses principes, sa conception, sa mise en oeuvre.

UN EXEMPLE D'UTILISATION DU MODE CONVERSATIONNEL :

LES MACHINES SIMULEES POUR L'ENSEIGNEMENT

DE LA PROGRAMMATION

Remarques sur les modes d'exploitation

Incidence sur les échanges du programme avec l'extérieur

Deux types d'exploitation sont généralement offerts par les systèmes actuels :

- Le mode de traitement en continu :
  Par abus de langage, ce mode est généralement appelé : traitement par trains. Cette convention sera donc appliquée dans la suite.
  Les programmes soumis à ce type d'exploitation effectuent leurs entrées-sorties généralement sur des supports à accès rapide (disques)
  Les données doivent être introduites avant l'exécution sur des files d'attente d'entrée. Les résultats sont sortis sur d'autres files d'attente qui sont vidées en fonction de la disponibilité des organes de sortie.
- Le mode de traitement intéractif :
  Chaque travail soumis à ce type d'exploitation dispose d'au moins un organe d'entrée et un organe de sortie pour communiquer avec l'extérieur. Les entrées peuvent donc être effectuées au moment où le programme demande des données, les résultats sont fournis dès leur mise en forme. Le programme dialogue donc avec l'usager, d'où le nom d'exploitation "conversationnelle".

Chacun de ces modes d'exploitation présente des avantages : l'un ou l'autre peut être préférable pour chaque type d'application.

Dans le cas de programmes destinés à l'enseignement de la
grammation, les fonctions de mise au point sont d'importan
primordiale. Or des possibilités très différentes sont off
dans ce domaine par les deux modes d'exploitation précéden

Mode de traitement par trains

La soumission d'un travail à ce mode de traitement suppose
l'acceptation de la désynchronisation totale des entrées e
des sorties par rapport à l'exécution du programme. Toutes
les données du programme devant être fournies dès la soumi
sion, et les résultats ne pouvant être obtenus qu'après ac
vement, les contrôles d'exécution, comme les autres donnée
doivent être prévus avant, et consultés après. Les possibi
offertes en mise au point sont donc les suivantes :

- copies instantanées de zones mémoire
  Ces copies peuvent être effectuées pendant l'exécuti
  en des points du programme choisis avant la soumissi
  du travail (SNAPS), ou relevées après son achèvement
  (POST MORTEM DUMPS).
  Les documents fournis par ces fonctions, bien que tr
  complets, sont généralement d'utilisation peu aisée.
  informations de toutes natures sont souvent mélangée
  et leur tri est toujours fastidieux, même si l'on en
  sage l'utilisation de programmes spéciaux d'analyse
  (qu'il convient alors d'écrire en fonction des infor
  tions à sélectionner).
  De plus, les points choisis pour la saisie de ces in
  tantanés ne sont pas toujours les bons : la lourdeur
  cette méthode ne permet pas de multiplier les points
  contrôle. De longues séquences peuvent être exécutée
  après une éventuelle anomalie, et le contexte d'une
  erreur peut alors être gravement altéré, voire total
  détruit avant la saisie.
- traces d'exécution
  Ce sont des séquences de programme, prévues lors de
  programmation, destinées à mémoriser une suite d'évi
  ments devant arriver lors de l'exécution du programm
  On peut mémoriser ainsi des passages à des points pr
  du programme, des valeurs de variables etc...
  Si ces traces permettent des points de contrôle plus
  nombreux que les copies de mémoire, la difficulté de
  recréer le contexte de l'anomalie demeure : une seule
  trace de ce type est rarement suffisante pour localiser
  une anomalie, et il faut alors rapprocher des informations
  figurant dans plusieurs traces.

Dans tous les cas, les renseignements fournis sur une anomalie éventuelle ne peuvent être exploités qu'après achèvement complet. S'ils s'avèrent insuffisants, une relance du travail est nécessaire, qui sera peut être de nouveau insuffisante. Or à chaque soumission, le travail reste un certain temps dans les files d'attente, augmentant d'autant le temps de mise au point du programme.

Mode de traitement conversationnel

Contrairement au mode de traitement par trains, le mode conversationnel permet une entrée de données et une sortie de résultats en temps réel. Les contrôles d'exécution, comme les autres données peuvent être demandés et consultés dynamiquement, pendant l'exécution du travail.
A condition que des séquences spécialement adaptées à la mise au point soient ajoutées au programme, des fonctions très diverses peuvent être offertes : [1]

- contrôle de l'avancement du programme
  Alors qu'en mode "traitement par trains", l'exécution d'un travail est inévitablement conduite jusqu'à son terme (normal ou anormal), l'usager du système conversationnel peut garder un contrôle permanent de l'exécution du programme. L'examen des résultats, au fur et à mesure de leur sortie, peut permettre de détecter la manifestation d'une anomalie bien avant sa reconnaissance par le système (toutes les anomalies ne sont d'ailleurs pas détectables par le système). Les fonctions d'arrêt du programme permettent alors à l'usager d'interrompre l'exécution avant une altération irrémédiable du contexte de l'anomalie.
  Les arrêts généralement utilisés sont de trois types :
  - arrêts "à la demande"
    l'exécution du programme est interrompue à la demande de l'usager, par exemple après détection

d'une anomalie inattendue.

- arrêts programmés (pièges)
  l'exécution du programme est interrompue lors d
  passage à un point prévu à l'avance. Ceci peut
  utilisé pour recréer rapidement le contrôle d'u
  anomalie. Bien sûr, de tels pièges peuvent être
  posés dynamiquement (à l'occasion d'un arrêt).
- arrêts systématiques (pas à pas)
  l'exécution est interrompue à chaque pas du pro
  me, par exemple pour étudier en détail le dérou
  ment d'une séquence suspecte.

Après un arrêt il est possible d'envisager une relanc
à un point quelconque du programme.

- accès à la mémoire du programme

Lors des arrêts du programme, l'usager peut accéder à
mémoire allouée à sa tâche. Comme il peut choisir les
rêts, il peut décider d'effectuer ces accès aux momen
adéquats ; il peut de plus accéder directement aux va
bles-clés, sans être encombré d'informations superflu
Ces accès peuvent être du type :

- consultation :
  pour vérifier le contenu de variables signifiat
  à un moment donné.
- modification :
  accompagnés d'une relance du programme, ils perm
  tent de juger du comportement du programme dans
  contexte différent.

Tous les renseignements nécessaires peuvent donc être
tenus au moment favorable, et sous une forme aisément
ploitable. Les essais nécessaires peuvent être effect
sans passer par une soumission différée du travail. L
temps matériellement nécessaire à la mise au point es
donc réduit au minimum.

Extension des services offerts par les simulateurs

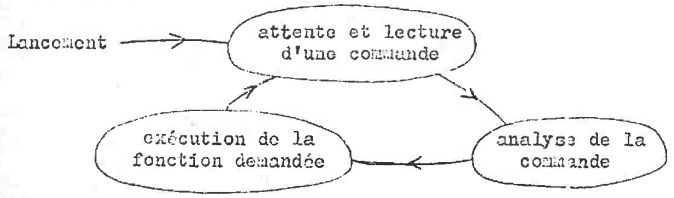
Principe de l'extension

Les "machines simulées" utilisées pour l'enseignement de la
programmation élémentaire sont des programmes permettant d'
cuter, sur un système réel, des programmes écrits par des é

diants conformément au code de machines imaginaires très simples.
Le programme de l'usager est fourni comme données à la "machine simulée". [II]
Grâce à la simplicité des machines choisies, l'étudiant n'a pas à affronter les difficultés inhérentes au système d'exploitation et à la complexité de la machine réelle.
Les simulateurs classiques sont conçus pour une exploitation par trains. Ils sont programmés pour offrir à l'étudiant des fonctions du type "unité centrale". Les possibilités de mise au point sont limitées aux copies de mémoire.
Si de nouvelles versions sont conçues pour une exploitation conversationnelle, des fonctions nouvelles peuvent être définies pour les machines imaginaires; en plus des fonctions programmées existantes, des fonctions du type "panneau de commande" peuvent être ajoutées. Des possibilités conversationnelles de mise au point sont dès lors disponibles. Elles permettent le contrôle et l'accès direct à la mémoire.

Fonctionnement du simulateur conversationnel

Pour l'usager, la version conversationnelle du simulateur doit réagir comme une machine réelle munie de son panneau de commande. Les différentes clés de la machine réelle son remplacées par des messages spéciaux frappés par l'usager. Le programme se comporte alors comme un interprèteur de ces commandes, qui réalise la boucle suivante :



Les fonctions offertes par ces commandes sont du type :

- accès aux organes de la machine (mémoire, registres, divers)
- contrôle des programmes (lancements, arrêts d'exécution)

Accès aux organes de la machine

Ces commandes permettent de consulter et de modifier les valeurs contenues dans les organes de mémorisation de la machine imaginaire.

Les commandes du type consultation accomplissent les fonc
suivantes :

- affichage d'un mot ou d'une zone de mémoire centrale
- affichage d'un registre de calcul
- affichage des registres de l'état-programme (compteu ordinal, codes-conditions, modes etc...)
- ...

Les commandes du type modification accomplissent les fonct
suivantes :

- modification du contenu d'un mot de mémoire centrale
- modification du contenu d'un registre de calcul
- modification du contenu d'un registre d'état-program
- ...

Utilisés pendant l'exécution simulée d'un programme usager
ces commandes permettent en particulier l'accès à toutes l
mémoires de ce programme.

Contrôle des programmes

Ces commandes permettent de charger le programme d'un usag
dans la mémoire de la machine imaginaire, de lancer, d'int
pre, de relancer l'exécution de ce programme au gré de l'u
Certaines commandes permettent de lancer l'exécution du pr
me simulé :

- en mode enchaîné (ceci est la seule fonction offerte la version du simulateur utilisée en mode de traitem par trains).
- pour une seule instruction du programme simulé (mode à pas).

Enfin des commandes permettent l'arrêt du programme simulé
les conditions suivantes :

- arrêt "à la demande" par frappe d'un caractère spéci pendant l'exécution du programme.
- arrêt programmé : des pièges peuvent être posés dans programme simulé.

Toutes les possibilités de mise au point offertes par le mo
conversationnel sont donc mises à la disposition de l'usag
sous la forme de ces commandes de simulation de panneau de
mande.

Caractéristiques d'exploitation des simulateurs conversationnels

Utilisation en sessions

Ces programmes sont utilisés par les étudiants au cours de séances de travaux pratiques. Pendant les sessions d'exploitation correspondantes, les usagers sont nombreux et le système doit donc gérer beaucoup de tâches simultanément. Pour garantir à chaque usager un temps de réponse minimum, le système doit assurer un partage équitable du temps d'unité centrale entre les différentes tâches.

Utilisation de programmes bien définis

Le système n'exécute que les simulateurs. En aucun cas un programme utilisateur n'est soumis : les essais des étudiants sont fournis comme données aux simulateurs. Les programmes contrôlés par le système sont donc présumés justes, les erreurs de l'usager étant filtrées par les simulateurs. Le système n'a donc pas à assurer de contrôle de validité sur les appels qu'il reçoit. Par ailleurs, les usagers n'ayant pas à écrire de programmes destinés à être directement contrôlés par le système, des règles strictes peuvent être imposées aux simulateurs. Aucune contrainte ne sera de ce fait imposée aux usagers.

Réentrance des programmes

Pendant la durée d'une session, les usagers sont nombreux, mais le nombre de programmes utilisés est très limité (les simulateurs) A un instant donné, plusieurs usagers du système utilisent donc généralement un même programme.
Pour une bonne exploitation, les programmes utilisés doivent être écrits réentrants, pour éviter au système de gérer plusieurs exemplaires d'un même programme. Cette restriction imposée aux simulateurs ne peut gêner les usagers.

Programmes effectuant beaucoup d'Entrées/Sorties

Les simulateurs exécutent beaucoup d'entrées-sorties sur le terminal : ces opérations sont demandées par le programme soit entre deux phases de travail (pour obtenir une commande du type "panneau de commandes" par exemple), soit pendant le déroulement d'un traitement logique (entrée-sortie demandée au cours de la simulation d'exécution d'un essai d'usager). Dans tous les cas

les sequences de calcul lancées entre deux accès au termi
sont courtes.
Or chaque entrée-sortie au terminal est une opération lon
en temps réel, pendant laquelle l'unité centrale peut êtr
utilisée pour le compte d'un autre usager. Les Entrées-So
au terminal s'accompagnent donc de commutations de tâches.
Ces accès étant nombreux, le partage du temps entre les us
ne nécessite pas de découpage du temps en tranches fixes.
rares séquences d'unité centrale longues existent dans le
gramme, elles peuvent être artificiellement coupées par l'
cution d'opérations "bidon" forçant la commutation. Cette
triction imposée aux simulateurs ne peut pas gêner les usa

Accès aux fichiers

Pendant une session, les usagers doivent pouvoir accéder à
nombreux petits fichiers en création, lecture, modificatio
destruction. Chaque usager doit pouvoir en utiliser un pet
nombre à un instant donné, mais aussi pouvoir en changer à
volonté.
Le système doit donc permettre ces accès aux simulateurs,
assurer la sauvegarde de ces fichiers pour une exploitatio
ultérieure éventuelle en traitement par trains.

ETUDE DES PRINCIPAUX SERVICES CONVERSATIONNELS OFFERTS PAR SIRIS 8

Le Sous-système Temps Partagé [III]

Ce sous-système permet à plusieurs usagers de demander simultanément l'exécution conversationnelle de n'importe quel programme. Naturellement, il se prête mieux à l'exploitation de programmes spécialement conçus pour une exécution en mode conversationnel.
Pour son utilisation, aucune hypothèse n'est émise, ni sur le type des applications qu'il gère, ni sur les usagers pour le compte desquels il travaille.

Utilisations quelconques

Aucune hypothèse n'étant émise sur les applications gérées, les travaux soumis peuvent être des processeurs communs préécrits, ou des programmes usagers.
Il en résulte que :

- des contraintes minimales doivent être imposées aux programmes pour causer un minimum de gêne aux usagers. Les seules contraintes théoriquement imposées sont celles concernant l'interface attendue lors des appels-système (la même que pour l'exploitation par trains).
- un contrôle de validité systématique doit être effectué lors de tous les appels-système : les programmes soumis ne peuvent être présumés sans erreurs.
- afin d'assurer un partage équitable de l'unité centrale entre les usagers, une allocation par tranches de temps est impérative, puisque des tâches faisant beaucoup de

calculs peuvent être soumises au même titre que des
effectuant beaucoup d'entrées-sorties.
- étant donnée la supposée diversité des applications
rées à un instant donné, les textes des programmes n
sont jamais partagés entre plusieurs tâches.

Utilisateurs quelconques

Aucune hypothèse n'étant émise sur les usagers , les trava
n'ont accès qu'à des supports disques communs et partageab
Les accès à ces disques ne comportent aucune restriction
rapport au mode de traitement par trains : création, lectu
modification, destruction de fichiers sont autorisées. De
les fichiers accédés en traitement par trains sont aussi a
bles en temps partagé, et réciproquement.
→ Ce sous-système, permettant l'exécution de tout programme
sationnel, peut donc être utilisé pour l'exploitation des s
lateurs ; mais le rendement de la machine n'est pas optima
il peut être amélioré par l'utilisation d'un système spéci
tenant compte des caractéristiques particulières d'exploit
des simulateurs.

Le Sous-système STRATEGE [IV]

Le système STRATEGE (Système de Transaction pour des Appli
tions en TElé-GEstion) est un système spécialisé permettan
contrôler des transactions soumises simultanément à parti
réseau de terminaux. Géré par SIRIS 8 comme une tâche Tem
(FUT), l'ensemble de ses usagers est vu par le système cent
comme un usager unique.

Utilisations spécialisées

Les fonctions offertes aux usagers sont exclusivement cel
contenues dans une bibliothèque de programmes du type : t
sactions. Ces fonctions sont définies au central ; les us
des terminaux ne peuvent jamais soumettre leurs propres p
mes.
Il en résulte que :
- de légères contraintes sont imposées aux programmes
ne causent aucune gêne aux usagers. En particulier,
textes des programmes étant communs à tous les usag

ils doivent être écrits réentrants, afin de permettre leur partage.
- les programmes gérés sont composés d'un enchaînement de phases de travail dont chacune est du type :
entrée de données ⟶ traitement ⟶ sortie des résultats
Ce découpage se prête mal aux traitements où l'entrée des données est demandée au cours d'une phase logique de travail.
- les programmes gérés ne peuvent accéder qu'à un ensemble limité de fichiers défini au lancement du système. Il n'est pas possible à une tâche de créer ni d'effacer directement des fichiers standard SIRIS 8.

Usager unique SIRIS 8

Le système complet étant considéré comme une tâche Temps Réel unique, les fichiers utilisés sont ceux accessibles par toute tâche de ce type. Ils peuvent être situés sur des supports disques quelconques.
Bien qu'il présente des caractéristiques communes avec le système idéal nécessaire à l'exploitation des simulateurs conversationnels, le système STRATEGE demeure mal adapté à cette utilisation, à cause du type très limitatif des programmes contrôlés, et des restrictions d'accès aux fichiers.

Systèmes de gestion de bases de données

Mentionnés ici pour mémoire, ils ne sont évidemment pas adaptés à l'exploitation des simulateurs conversationnels.

La Méthode d'accès TAM Mode Caractère [V]

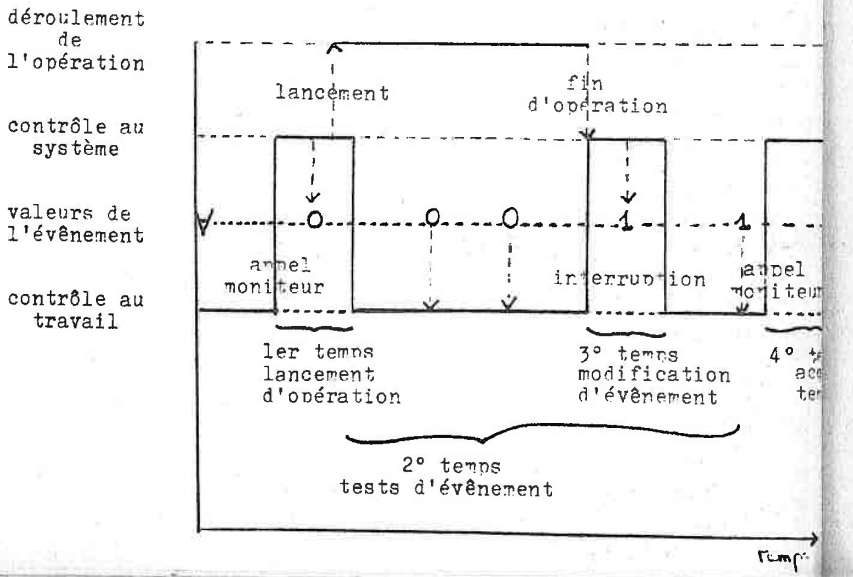
C'est un ensemble d'appels-moniteur utilisables dans un programme standard et permettant à ce programme d'effectuer des entrées-sorties sur un ou plusieurs terminaux en Mode Caractère.

Principe

Des appels-moniteur permettent d'exécuter des opérations de lecture ou d'écriture sur un terminal. Ces opérations sont asynchro-

nes ; elles se déroulent en plusieurs temps :

- 1er temps : lancement de l'opération par appel-moniteur exécuté par le travail.
  L'entrée-sortie sur le terminal commence, mais le travail qui l'a demandée garde le contrôle de l'unité centrale, et peut continuer à s'exécuter. Une information spéciale d'événement est alors initialisée par le système. Elle sera modifiée par le système à l'achèvement de l'opération physique (ECB).
- 2° temps : tests éventuels de l'information d'événement. Pendant le déroulement de l'opération physique d'accès au terminal, le programme peut tester périodiquement cette information pour se tenir au courant de l'achèvement de l'opération.
- 3° temps : achèvement de l'opération physique.
  Le système modifie l'information d'événement afin de prévenir le programme. Ce dernier n'intervient pas dans cette opération.
- 4° temps : acquittement de l'opération.
  Un appel-moniteur exécuté par le travail acquitte l'opération et permet d'informer le programme des conditions de déroulement de l'opération (anomalies dues à la transmission de caractères spéciaux, détection de coupures de ligne...) par fourniture d'un code d'achèvement.

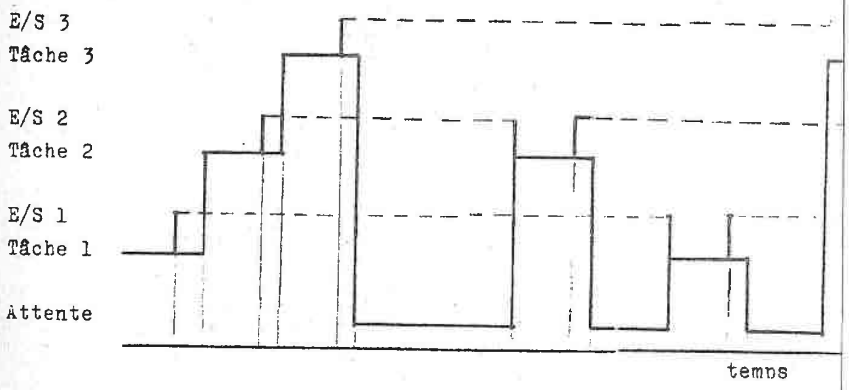


Utilisation de l'Asynchronisme

L'opération d'entrée-sortie sur un terminal peut être très longue. Pendant cette opération, le travail demandeur garde le contrôle de l'unité centrale. Il peut alors :

- soit continuer à travailler, et donc lancer une autre opération sur un autre terminal, voire sur le même (les demandes sur un même terminal sont automatiquement mises en file d'attente).
- soit se mettre en attente de l'arrivée d'un ou plusieurs événements par exécution d'un appel-moniteur. Il perd alors le contrôle de l'unité centrale et le retrouvera à l'arrivée du ou des événements attendus, si sa priorité dans le contexte de multiprogrammation de SIRIS 8 le permet.

Cette possibilité de lancer des opérations simultanées sur plusieurs terminaux permet au travail d'exécuter des opérations pour le compte de plusieurs terminaux simultanément, la tâche associée à un terminal pouvant être traitée pendant les entrées-sorties des autres. Si les demandes sont simples, et le nombre des terminaux restreint, aucun ralentissement ne sera sensible, comme le montre le schéma suivant :



Les traits pleins horizontaux représentent le temps d'unité centrale utilisé par les tâches associées à chaque terminal, les traits pointillés représentent les temps d'exécution des Entrées-Sorties physiques sur les terminaux.

Surveillance du Terminal

C'est une opération de lecture particulière qui peut être lancée

en toutes circonstances, que le terminal soit opérationnel
non, et même en parallèle avec une autre opération sur le m
terminal. Elle permet par polling de détecter l'émission pa
terminal de caractères spéciaux (dits : Attentions).
Une utilisation opportune d'une telle opération peut permet
au programme qui la lance

- d'être activé même si les terminaux ne sont pas opé
  tionnels, et de ne commencer le traitement que lorsqu
  signalent leur mise en service.
- de détecter les demandes asynchrones lancées par l'us
  du terminal pendant l'exécution du travail, alors qu'
  cune opération réelle de lecture n'est en cours sur c
  minal.

⟶ Un programme du type temps réel utilisant cette méthode d'a
peut donc offrir des services conversationnels à un termina
à un réseau de terminaux, sous réserve qu'il gère lui même
partage des ressources qu'il offre, entre les terminaux de
réseau.
Cette gestion étant totalement assumée par le programme, le
algorithmes choisis peuvent être optimaux pour l'applicatio
Par ailleurs, tous les services d'accès aux fichiers offert
par SIRIS 8 sont accessibles à un programme de ce type ; le
simulateurs peuvent donc être exploités de façon optimale s
utilisent ces services ; mais un sous-système conversationn
doit alors être adjoint au texte des simulateurs eux-mêmes
gérer le partage des services offerts aux usagers.

SPECIFICATIONS EXTERNES DU MONITEUR SAFRAN

Principe du moniteur

Relation moniteur-application

Le moniteur SAFRAN est un noyau de système permettant la mise en oeuvre simple et peu coûteuse d'applications conversationnelles à plusieurs points d'accès présentant des caractéristiques d'exploitation analogues à celles des machines simulées.
Pour celà, il prend en charge lui-même les fonctions de gestion des terminaux et de partage des ressources entre les différents usagers. Le concepteur des applications n'a donc plus à réaliser que la partie de logiciel propre à l'application elle-même : toutes les opérations concernant les terminaux et les interactions entre usagers sont traitées, dans le logiciel d'application, par des appels aux services SAFRAN.
Le système SAFRAN se présente donc sous la forme d'un ensemble de services qui, regroupés avec le logiciel propre à l'application (un ou plusieurs programmes respectant une interface définie) forment une unique tâche temps réel SIRIS 8. Les accès aux terminaux sont traités par des appels à la méthode d'accès TAM.

Rôles du moniteur [VI]

Le rôle tenu par le moniteur SAFRAN est comparable à celui d'un moniteur temps partagé. Il assure :

- la gestion d'un dialogue avec l'opérateur central.
  Bien que nécessaire pour d'évidentes raisons d'exploitation, ce dialogue doit cependant rester discret pour ne pas gêner l'exploitation.
- la gestion du dialogue avec les terminaux.
  Ce dialogue doit être aussi naturel que possible, pour

deux raisons essentielles :
Il doit être transparent pour les usagers des terminaux, seule la logique de l'application doit être perçue.
Il doit être manipulable par des usagers non nécessairem familiers des techniques informatiques.

- la gestion du dialogue avec l'application proprement dite ce dialogue doit être riche, mais il ne doit imposer qu'u minimum de contraintes à l'application, afin de ne pas co pliquer sa réalisation.
- le partage des ressources entre les usagers, afin de réal ser une apparente simultanéité des traitements associés à chaque point d'accès, et garantir de ce fait un temps de réponse minimal.

Philosophie de conception

Afin d'assurer une mise en oeuvre simple et une utilisation sou la réalisation du système SAFRAN doit respecter des règles supp mentaires :

- une application contrôlée par le système SAFRAN doit être exécutable sous le contrôle d'un moniteur SIRIS 8 standar Aucune modification spéciale ne doit être introduite dans le système officiel, aucune génération de système particulière ne doit être nécessaire (sous réserve que les resso ces standard nécessaires aient été incluses dans la versi courante) (modules SIRIS 8 d'accès aux terminaux etc...).
- une application SAFRAN étant un programme standard, elle nécessite la présence d'un noyau de programme en mémoire pendant toute la durée de la session. L'encombrement de ce noyau doit être limité au maximum, pour ne pas pénalis l'exploitation centrale.
- une application SAFRAN doit avoir accès à un maximum des services offerts par SIRIS 8 pour ne pas être défavorisée par rapport à une application temps partagé.
- une application SAFRAN est généralement fort différente d'une autre :
le réseau de terminaux peut être différent en nombre et e type, les services offerts peuvent nécessiter plus ou moi de ressources, les fichiers utilisés peuvent avoir des ca téristiques différentes etc... Le même logiciel doit être utilisable dans tous les cas, et doit donc pouvoir être ad té à toute configuration logicielle ou matérielle d'appli tion.
- certaines applications peuvent nécessiter l addition de services non prévus dans la définition standard du système SAFRAN. La possibilité de telles additions doit être autorisée, de façon à ne pas entraîner, dans la majorité des cas, de modifications du logiciel standard, tout en permettant l'accès aux fonctions de base. Si une telle modification était cependant nécessaire, elle doit être facilitée par une conception modulaire du système et par une disposition claire et très commentée du programme source.

Résumé des principales fonctions

Fonctions offertes à l'opérateur central

- une application SAFRAN peut être lancée à partir du poste de contrôle de la machine, comme un travail ordinaire, par une commande standard SIRIS 8 : P (lancement d'un travail parallèle (UT) ou F (lancement d'un travail temps réel (FUT)), sous réserve de la création préalable de la procédure cataloguée correspondante.
- une telle application, comme un travail ordinaire, peut être abandonnée en catastrophe par une commande standard X (avortement opérateur) lancée au poste de contrôle de la machine. La tâche est alors abandonnée avec le code E OF.
- cette méthode d'arrêt en catastrophe doit être évitée dans la mesure du possible, car elle provoque l'arrêt immédiat des dialogues avec les terminaux et la perte des fichiers en cours de création. Un arrêt en douceur peut être demandé à partir du poste de contrôle, par une commande standard I (transmission d'une interruption opérateur). Après réception d'un tel signal, toute nouvelle demande de travail est refusée, et l'application s'arrête d'elle même après achèvement de la dernière opération en cours.
- afin d'informer l'opérateur central de l'état de charge du système, toute application SAFRAN imprime le nombre de terminaux connectés, sur le poste de commande de la machine, et ce à chaque prise de contact, et à chaque déconnexion d'un terminal.

Fonctions offertes aux usagers des terminaux (services SAFRAN)

- toute demande de services d'un usager à une application doit être précédée d'une demande de connexion, permettant au système SAFRAN d'initialiser une tâche pour le compte d cet usager. Un indicatif associé à l'usager est alors créé par SAFRAN, puis est transmis au terminal.
  Même lorsque l'application est active, une demande de connexion peut être rejetée si un arrêt en douceur a été demand par l'opérateur central.
- après une demande de connexion acceptée par l'application, l'usager peut demander l'exécution d'un des programmes spé fiques de l'application. Ces programmes sont appelés servi d'application. Le dialogue entre l'application et l'usager est alors commandé par la logique propre à l'application. Un nouveau service pourra être demandé après l'achèvement du service en cours.
- l'usager peut à son gré transmettre des signaux asynchrone à l'application même si aucun dialogue n'est en cours avec elle. Ces signaux asynchrones sont les seules formes d'éch ges à l'initiative de l'usager.
- pour mémoire, un service standard du système SAFRAN permet à un usager d'envoyer un message à un autre usager. Ce mes ge sera reçu par le correspondant après la fin du service en cours pour le compte du correspondant.
- une session de travail d'un usager avec une application es normalement interrompue dans un des cas suivants :
  soit à l'initiative de l'usager qui transmet à l'applicati une demande de déconnexion après la fin d'exécution de la dernière demande de service. Une nouvelle connexion ultéri est alors possible.
  soit à l'initiative du système après une demande d'arrêt en douceur émise par l'opérateur central. L'usager est alo informé de l'arrêt d'exploitation, toute nouvelle demande connexion ultérieure est alors refusée.

Fonctions ouvertes au logiciel d'application (fonctions SAFR

Le logiciel spécifique de l'application est divisé en un ou plusieurs services appelables indépendamment par chaque usag connecté. L'exécution de ces services est lancée par le moni SAFRAN à la suite d'une demande émise par un usager.
Ce logiciel a accès :

- à toutes les instructions standard de l'IRIS 80.
- éventuellement à certaines instructions privilégiées si l'application dispose de l'accès au mode maître.
- à la majorité des services offerts par le moniteur SIRIS 8.
- à toutes les fonctions standard offertes par le moniteur SAFRAN
- à d'éventuelles fonctions non standard ajoutées au moniteur pour l'application considérée (ces services ajoutés ont accès à tous les services standard du moniteur SIRIS 8. L'accès aux services privilégiés leur est ouvert si l'application est du type FUT (lancé par une commande F au poste de contrôle)).

Les fonctions suivantes sont offertes au logiciel d'application par le moniteur SAFRAN standard :

- accès au terminal en lecture ou en écriture.
  Ces opérations sont synchrones pour le logiciel d'application : l'accès au terminal provoque l'arrêt du travail pour le compte de l'usager concerné. Celui-ci ne reprend qu'après l'achèvement de l'opération d'entrée-sortie associée. Ces accès aux terminaux participent donc au partage du temps d'unité centrale entre les usagers de l'application, puisque chaque accès provoque la commutation.
- forçage de la commutation des tâches.
  Cette opération permet à un service du logiciel d'application d'arrêter temporairement son travail pour le compte de la demande usager en cours de satisfaction, lorsqu'aucun accès au terminal provoquant une commutation de tâches n'est prévu dans un délai raisonnable. Ces services, associés aux accès des terminaux, permettent un partage équitable du temps d'unité centrale entre les usagers, même lorsque des services faisant beaucoup de calculs sont offerts par l'application. L'allocation du temps par tranches est donc utile.
- demande d'ajout ou de non-ajout des caractères spéciaux provoquant le retour à la ligne en fin des messages émis vers le terminal lors des accès en écriture.
  Ces fonctions peuvent être importantes en cas d'édition de textes sur les terminaux (possibilité d'écrire une ligne en plusieurs accès) ou en cas de dialogues importants (possibilité d'envoyer l'écho de la réponse sur la même ligne que la question, présentation plus naturelle). Par défaut, le retour à la ligne est implicite en fin de message.

- synchronisation entre demandes en provenance de plus
usagers. Des services demandés par des usagers diffé
peuvent ainsi s'attendre, ou s'introduire seuls dans
séquences exclusives. Cette synchronisation est obte
par des fonctions d'accès à des informations du type
sémaphores.
- obtention des codes d'anomalies détectées par le mon
SAFRAN. Les anomalies peuvent être dues à :
des demandes impossibles transmises au moniteur.
des incidents sur la liaison avec le terminal.
la détection de signaux asynchrones lancés par l'usa
Ces codes d'anomalies sont décrits en annexe.
- obtention de l'indicatif de l'usager associé.
Cette information permet au logiciel d'application d
connaître pour le compte de quel usager il travaille.
- demande d'enchaînement automatique de services.
A la fin de l'exécution du service d'application en c
le contrôle est normalement rendu au moniteur, l'usa
pouvant alors demander l'exécution d'un nouveau serv
Cette demande permet d'éviter le retour du contrôle a
moniteur en fin d'exécution du service en cours, et d
forcer l'enchaînement automatique d'un autre service,
intervention de l'usager.
- fin d'exécution du service d'application en cours.
Si aucune demande d'enchaînement automatique n'a été
mise au système, le contrôle est rendu au moniteur, a
l'exécution du service demandé est entreprise pour le
du même usager.

Les règles précises d'appel à ces fonctions sont décrites e
annexe A.

PRINCIPES GENERAUX DU SYSTEME SAFRAN

Traitement des requêtes des usagers

Notion de tâche usager [VI]

Une application SAFRAN connaît autant d'usagers qu'il y a de terminaux susceptibles de demander une connexion à cette application : à chaque terminal est associé un usager unique.
Un usager demande un travail à une application SAFRAN sous la forme d'une demande de connexion de son terminal, suivie d'une suite de demandes de services standard ou spécifiques à l'application, et achève le travail par une demande de déconnexion.
Chaque service spécifique en cours d'exécution pour le compte d'un usager lance des requêtes au système SAFRAN sous la forme de demandes d'exécution de fonctions offertes au logiciel d'application.
On appelle tâche usager la séquence de programme exécutée par l'ordinateur pour la satisfaction de toutes les requêtes en provenance d'un terminal, pendant toute la durée de la session.
Une tâche usager se compose donc de séquences du moniteur SAFRAN et de séquences du logiciel d'application.

Evolution de la tâche usager

Pour satisfaire les demandes de services lancées par l'usager, la tâche usager passe successivement dans des séquences de programme et dans des états d'attente logique :

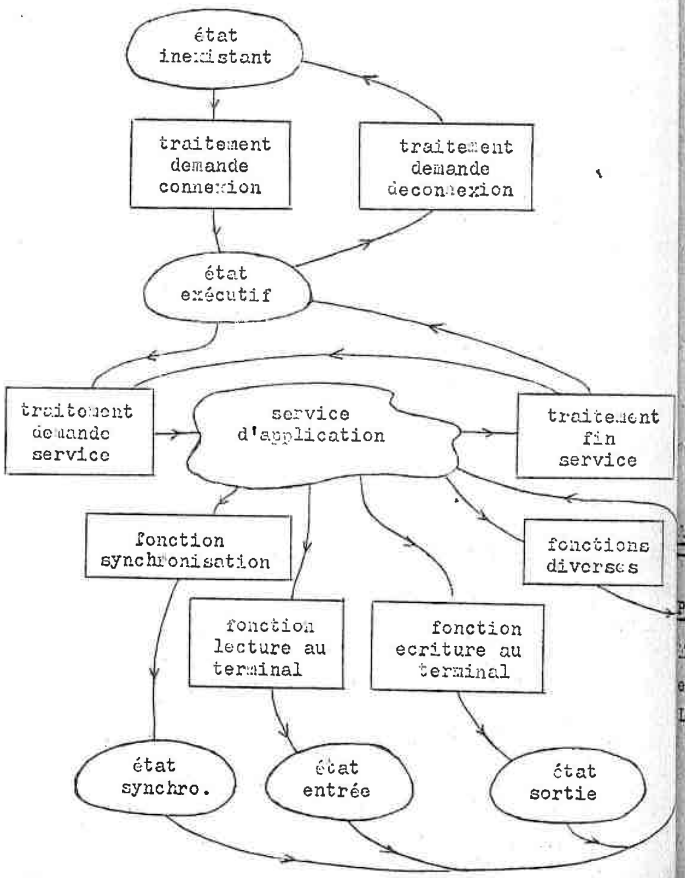
- l'état initial de la tâche est celui où le terminal n'est pas connecté à l'application. La tâche est dite inexistante.
- une commande de connexion lancée par l'usager provoque la création de la tâche et son passage à l'état dit : exécutif. La transition inverse est provoquée par une demande de déconnection.
- dans l'état exécutif, la tâche est en attente. Elle est

prête à recevoir de l'usager et à traiter une demande de service standard ou spécifique d'application.
En fin d'exécution du service, la tâche repasse à l'état exécutif, si aucune demande d'enchaînement n'a été lancée.

- pendant l'exécution d'un service d'application, des fonctions SAFRAN peuvent être demandées par le logiciel d'application. La plupart de ces fonctions sont immédiatement traitées, mais certaines provoquent la mise en attente logique de la tâche (entrée au terminal, sortie au terminal, synchronisation entre tâches ...).

Le schéma suivant décrit l'évolution de la tâche :

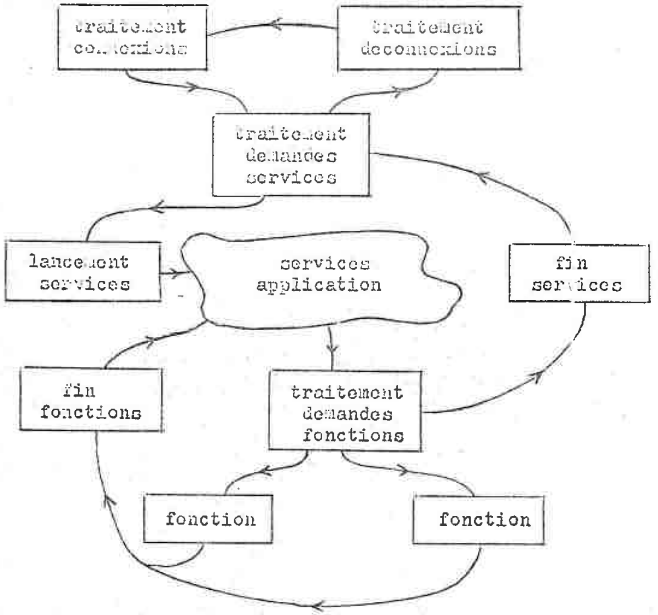
- Les pavés ovales correspondent à des états d'attente logique.
- Les pavés rectangulaires correspondent à des séquences du moniteur.
- Les patates correspondent à des séquences d'application.



Architecture logique de la tâche usager

Une tâche usager se compose donc de l'enchaînement logique de traitements dont certains sont interrompus temporairement par des opérations bloquantes (synchronisation, accès aux terminaux).
Le schéma suivant est une description simplifiée de l'enchaînement des séquences de programme constituant la tâche usager :

- les pavés rectangulaires correspondent à des séquences du système.
- les patates correspondent au logiciel d'application.

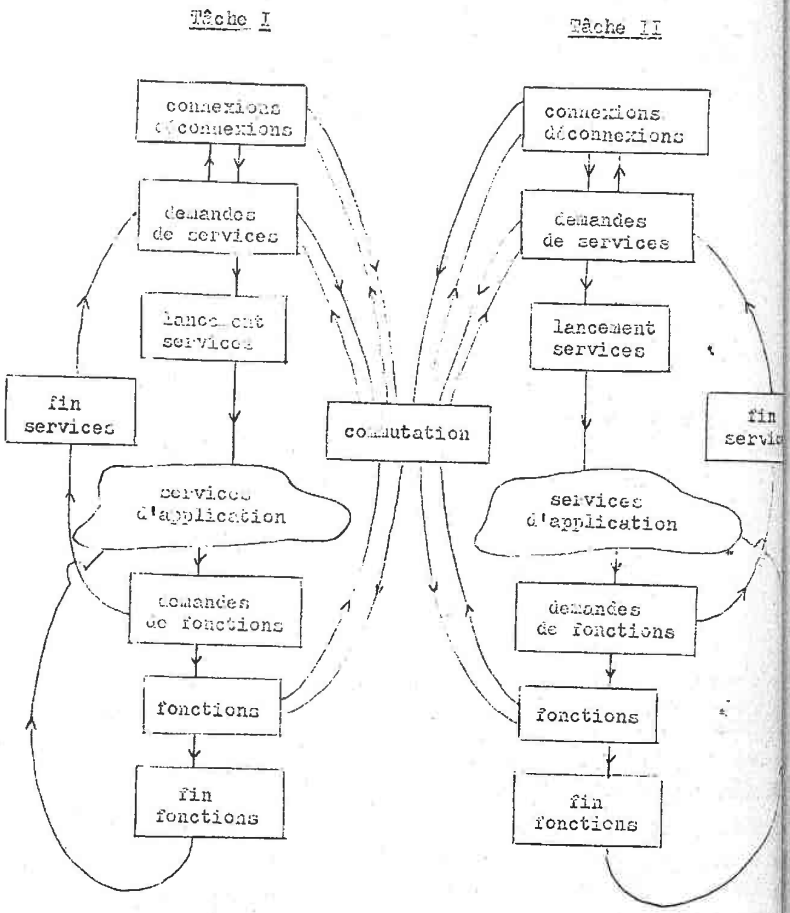


Architecture générale du système

Pseudo-simultanéité des tâches-usagers

En pratique, plusieurs usagers veulent contrôler une tâche de ce type en simultanéité apparente.
L'exécution d'une tâche n'est donc pas physiquement poursuivie

de façon continue, d'autant que, par hypothèse, elle passe souvent par des états d'attente dûs aux accès au terminal (travail conversationnel). Chaque état d'attente logique de la tâche se traduit par interruption de l'exécution au profit d'une autre tâche si l'attente logique de cette nouvelle tâche est alors terminée. L'architecture logique de l'application, dans le cas de deux tâches peut donc être décrite par le schéma suivant :



Réentrance du texte de l'application

Une même application mettant à la disposition de tous les usagers des services identiques, les séquences de programmes exécutables par chaque tâche sont identiques. Il n'est donc pas nécessaire d'avoir en mémoire autant d'exemplaires du même texte qu'il y a de tâches gérables. Le texte du programme est donc commun à toutes les tâches. Seules, les variables existent à raison d'un exemplaire par tâche, car elles sont généralement différentes pour chaque tâche à un moment donné.
Cette réentrance du texte du programme est assurée de façons différentes pour le texte du système et pour le logiciel d'application.

Réentrance du texte du moniteur [VII]

Les séquences de programme composant le texte du système étant, en grande majorité, d'utilisation courante, le texte du moniteur fait partie du noyau résident de l'application.
De même, les variables de travail du moniteur, très peu nombreuses, font partie du noyau résident.
Ces variables de travail sont de deux types :

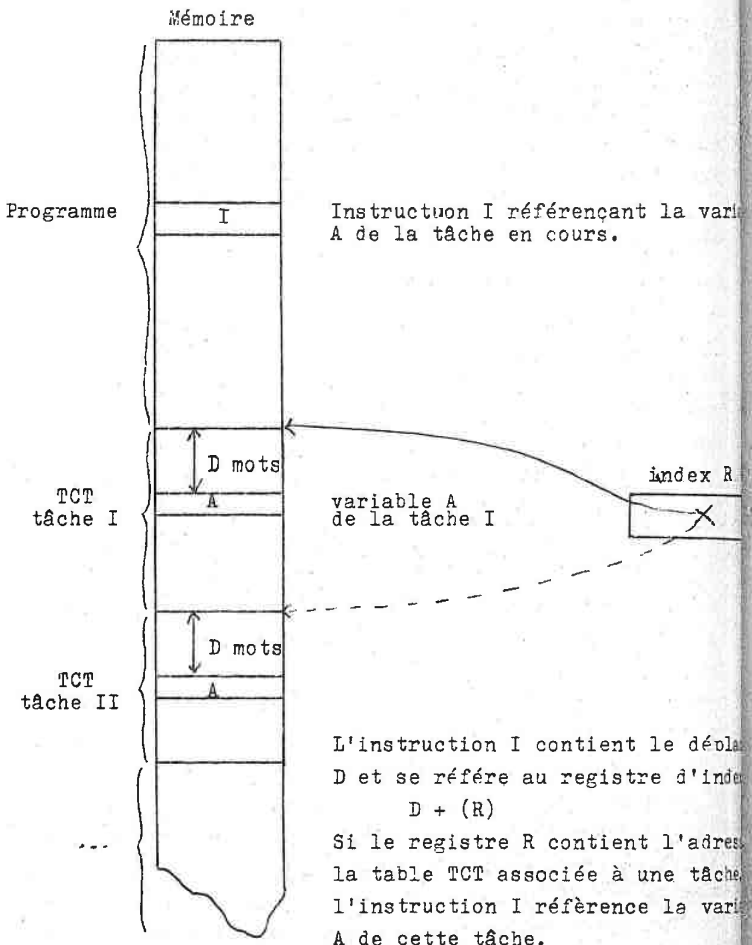
- les variables communes du système existent en un seul exemplaire.
- les variables propres de chaque tâche existent en autant d'exemplaires qu'il y a de tâches gérables.

Un même texte de programme doit pouvoir travailler sans modifications pour le compte de plusieurs tâches. Si l'adressage des variables communes ne pose aucun problème, l'accès aux variables propres ne peut s'effectuer par adressage direct simple : l'adresse d'une telle variable, fonction de la tâche en cours d'exécution, ne peut figurer dans le programme.
Le principe adopté pour l'accès à ces variables est le suivant :
Tout le contexte de chaque tâche (certaines variables et les pointeurs vers les autres variables) est regroupé dans des tables identiques appelées Tables des Contextes des Tâches (TCT).
Le déplacement d'une variable donnée par rapport au début de la table est le même pour toutes les tables.
L'accès aux variables peut alors être réalisé par des instructions indexées :

- la zone "déplacement" de l'instruction contient le déplacement de la variable par rapport au début de la table (le même quelle que soit la tâche).

- le registre d'index sert de base. Il doit contenir l'a
  se de la table.

La même instruction peut ainsi travailler pour le comnte de
tes les tâches, comme le montre le schéma suivant :



Cas du logiciel d'application [VI] [VIII]

Les séquences de programme constituant le logiciel d'application sont généralement assez longues et nombreuses. Il peut être souhaitable, pour limiter l'encombrement mémoire de l'application, de ne pas les conserver dans le noyau résident.
De même, les variables de travail de ces séquences ne sont généralement pas très réduites, et si le nombre de tâches à gérer est grand, l'espace occupé par la totalité des variables pourrait devenir très conséquent si elles étaient toutes résidentes.
Pour permettre une non-résidence du logiciel d'application, une application SAFRAN doit contenir deux espaces de mémoire libre suffisants pour stocker la plus longue des séquences d'application d'une part, et la plus longue des zones de variables utilisées par le logiciel d'application d'autre part.
Les textes des programmes d'application sont alors stockés dès la conception de l'application dans un fichier sur mémoire externe (disque) et sont chargés dans la zone mémoire libre par le moniteur SAFRAN toutes les fois que leur présence est nécessaire en mémoire (swapping IN).
Pour les zones de variables des programmes d'application, un autre fichier sur mémoire externe est utilisé par l'application, et contient, pour chaque tâche gérable, un espace égal à la plus longue des zones de variables utilisées par les programmes d'application.
Lorsqu'une tâche usager n'est pas physiquement active, la zone des variables correspondante est copiée sur ce support externe. Elle est introduite en mémoire libre avant le passage en état actif, et sauvegardée sur la mémoire externe à la désactivation. (swapping IN et OUT).

Avantages et inconvénients

Un tel système permet, outre le gain de place, d'exécuter les séquences du logiciel d'application à un emplacement fixe en mémoire, et avec des variables situées toujours à la même adresse, quelle que soit la tâche active. Le partage de ces textes peut donc être assuré sans aucune restriction quant aux modes d'adressage utilisés dans la programmation.
Par contre, la présence d'un seul service d'application en mémoire à un instant donné a les inconvénients suivants :

- les chargements de textes sont fréquents.
- les swappings de zones de variables sont systématiques.

- l'application entière est généralement en attente pend
les swappings.

Une version ultérieure du moniteur SAFRAN pourrait améliorer cette situation en permettant une présence simultanée en mém de plusieurs services :

- le nombre de swappings serait diminué par l'augmentatio de la probabilité de présence en mémoire des zones néce saires.
- certaines tâches pourraient être actives pendant les sw pings des autres.

Un tel système imposerait des contraintes de programmation au services d'application afin de permettre une relocation des t tes et des variables : l'action sur la transformation d'adres virtuelle en adresse réelle n'étant pas un service standard SIRIS 8, il faudrait avoir recours à un "tricolage" probable dépendant des versions du système SIRIS 8, et donc de mainten lourde.

Les solutions envisageables seraient les suivantes :

- tous les accès mémoire pourraient se référer à des regi tres d'index (suivant le même principe que le texte du système).
- une programmation complète (moniteur et logiciel d'appl cation) en mode C répondrait aussi à la question, mais toutes les instructions avec accès mémoire devraient se référer à des registres de base.

Dans tous les cas, la manipulation des adresses serait très délicate.

Organisation du système

Principe

Une application SAFRAN est donc formée de deux parties princi pales :

- les traitements de la tâche usager, texte formé des sé quences réentrantes du moniteur et des logiciels d'appl cation gérés par swapping.
- la séquence de commutation des tâches, appelée en plu sieurs points de la tâche usager, et assurant la répart tion de l'unité centrale entre les usagers. Cette séque ce est aussi appelée : répartiteur.

La séquence de commutation des tâches est le centre vital du système, mais c'est une séquence simple. La logique de la tâche usager étant plus complexe, elle doit être divisée à des fins de modularité.

Notion de niveau

Les séquences de programme constituant la tâche usager peuvent se diviser en deux catégories :

- celles qui réalisent des opérations toujours séquentielles.
- celles qui peuvent être appelées en plusieurs points du système (sous-programmes).

Des séquences des deux catégories peuvent appeler la séquence de commutation des tâches. Par exemple, celle traitant les lectures au terminal est appelée de plusieurs points.
Or cette séquence provoque la mise en attente de la tâche pendant l'opération de lecture. Elle provoque donc une commutation de tâches.
La pile contenant la trace des appels successifs de sous-programmes pour le compte d'une tâche doit donc être conservée lors de chaque commutation, les tâches n'étant pas généralement dans le même état à un instant donné. Une pile existe donc pour chaque tâche. Son adresse est contenue dans le TCT associé à la tâche.

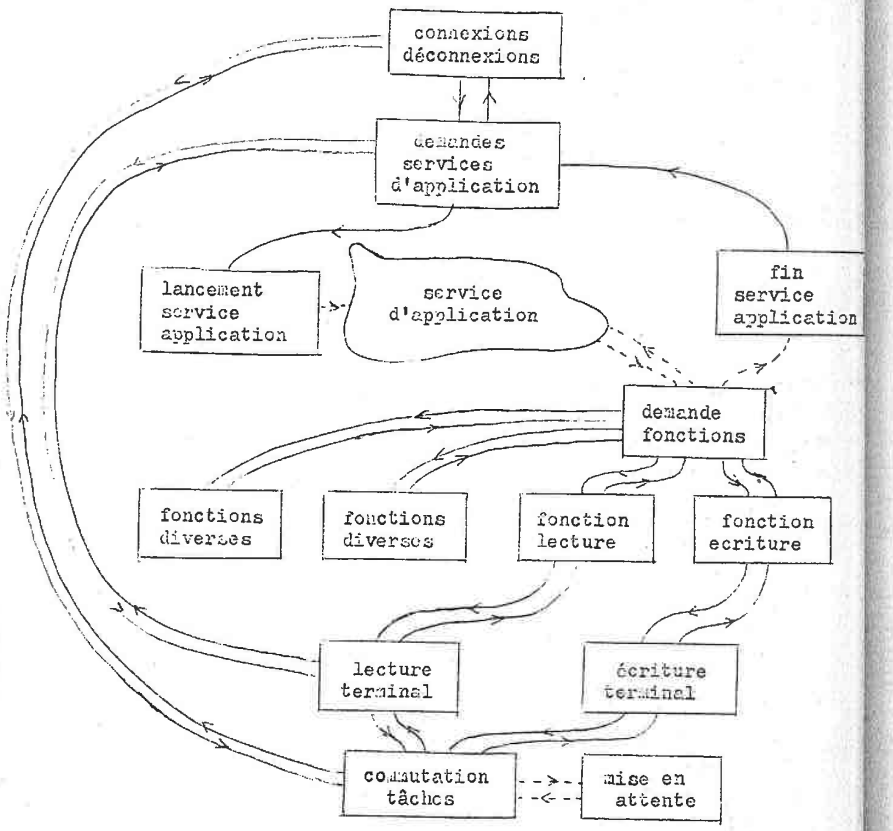
- Afin d'assurer une plus grande modularité, un maximum des traitements séquentiels ont artificiellement été mis sous forme de "sous-programmes". La majorité d'entre eux assurent chacun une action ponctuelle bien définie. Mais certains traitements sont en fait constitués de séquences indépendantes (le logiciel d'application et son interface, par exemple), et le nom de sous-programme peut difficilement leur être donné. Dans la description du système, seront désignées sous le nom de : niveau, toutes les séquences dont l'appel s'accompagne de l'empilement de l'adresse d'appel, et dont le retour s'effectue à l'adresse contenue dans le sommet de la pile des appels. Les niveaux peuvent être des sous-programmes naturels ou artificiellement constitués.

N.B. La séquence de commutation est traitée comme un niveau. Elle sera toujours la plus haute dans la pile d'une tâche, lorsqu'elle y figurera. Son fonctionnement sera décrit ultérieurement.

Organisation du système par niveaux

Le schéma suivant décrit une organisation simplifiée des traitements du système (tous les niveaux ne sont pas représentés. Voir description exacte en annexe C) :

- les flèches pointillées décrivent des enchaînements de traitemen dans le corps d'un même niveau.
- les flèches pleines décrivent des appels et des retours de nive



Remarque : Une séquence d'application avec l'interface de début, l'interface de fin, l'interface d'appel de fonctions constitue un niveau, au même titre que le sous-programme "lecture au terminal", par exemple.

FONCTIONNEMENT DU MONITEUR SAFRAN

(services communs)

Commutation des tâches

Principe de la commutation

Effectuer une commutation de tâches, c'est exécuter séquentiellement les opérations suivantes :

- interrompre l'exécution de la tâche active en sauvegardant l'adresse d'arrêt.
- choisir une tâche parmi celles qui ne sont pas en attente logique (tâches candidates à l'activation).
- relancer l'exécution de la tâche choisie à l adresse où elle avait été précédemment interrompue.

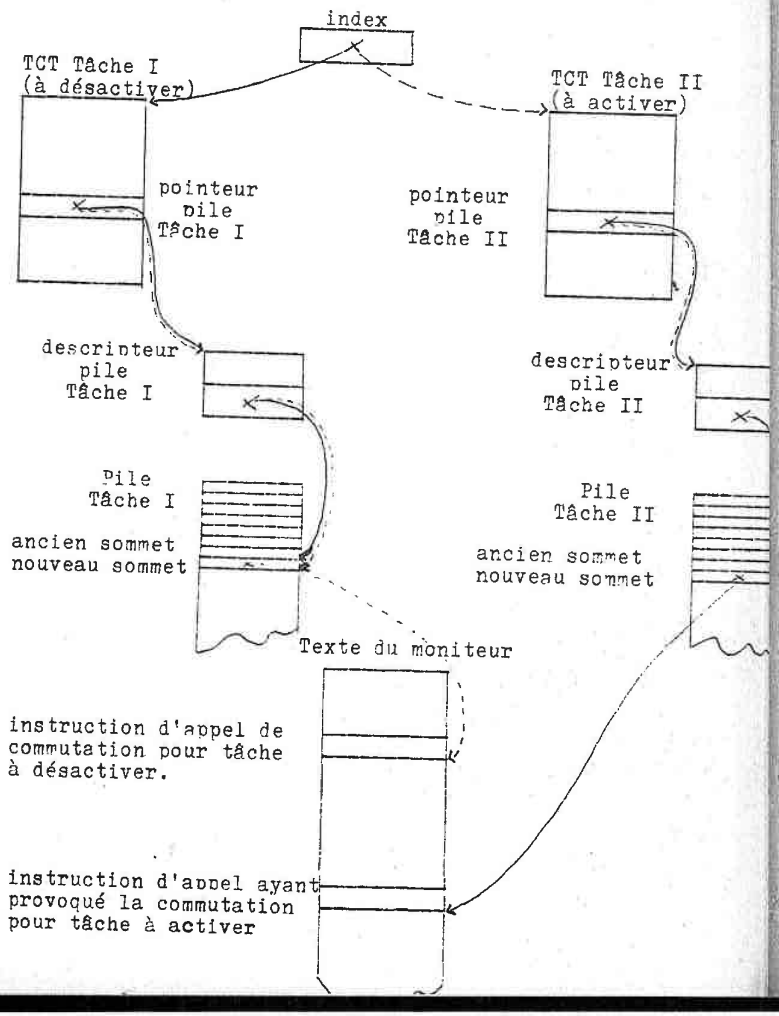
La commutation est toujours provoquée par un appel de la tâche active à la séquence de commutation : par hypothèse, les demandes volontaires de commutation lancées par les tâches sont assez fréquentes pour justifier l'absence de partage de temps en tranches fixes (time slicing). La séquence de commutation est appelée comme un niveau standard : le registre référencé par l'instruction d'appel (BAL) contient donc l'adresse de la première instruction non encore exécutée. Cette adresse peut être empilée sur la pile des appels de la tâche active (l'adresse de cette pile est contenue dans la table du contexte de la tâche active - TCT - dont l'adressse est contenue dans le registre index utilisé pour la réentrance).

Dès lors, la tâche est considérée comme désactivée, le point d'arrêt a été mémorisé. La séquence de commutation des tâches peut alors choisir une tâche candidate en fonction d'un algorithme qui sera décrit plus loin.

La tâche à activer ayant été choisie, l'adresse de la table TCT de cette nouvelle tâche est placée dans le registre index

utilisé pour la réentrance. La pile des appels de la nouvel
tâche peut ainsi être accédée. Son sommet contient l'adress
du point d'arrêt d'exécution sauvegardé à la désactivation.
L'activation de la nouvelle tâche se fait par simple retour
standard de niveau : l'adresse de retour est extraite au so
de la pile des appels.
Le schéma suivant visualise l'opération de commutation :

- les flèches pleines représentent les chaînages avant
  mutation.
- les flèches pointillées représentent les chaînages ap
  commutation.



Caractérisation des tâches candidates à l'activation [VI]

Par hypothèse, une tâche peut demander la commutation pour deux raisons :

- parce qu'elle se place en état d'attente logique (attente d'entrée ou de sortie au terminal, attente de synchronisation).
- parce qu'elle exécute une séquence de calcul trop longue et qu'elle décide de s'interrompre temporairement pour assurer une répartition équitable du temps de calcul.

Dans le deuxième cas, la tâche est immédiatement candidate à la réactivation. Dans le premier cas, elle ne sera candidate qu'après achèvement de l'attente logique.
La séquence de commutation des tâches doit choisir la tâche à activer parmi les tâches candidates. Elle doit donc avoir accès, pour chaque tâche inactive, aux renseignements suivants :

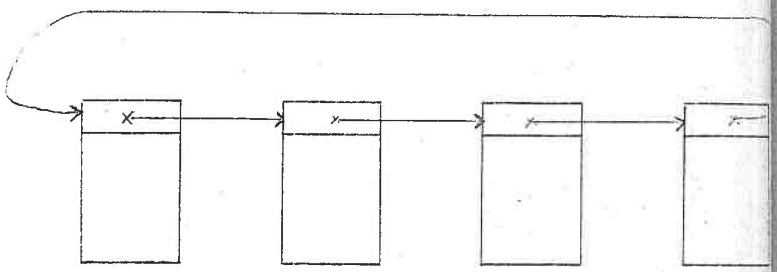
- variable logique indiquant si la tâche est candidate à la réactivation sans délai, ou si la réactivation est conditionnée par la fin d'une attente logique.
- dans le deuxième cas seulement, information d'événement indiquant la fin de l'attente logique.

Ces deux informations sont disponibles dans la table TCT de chaque tâche.

- la variable logique doit avoir été initialisée avant la demande de commutation, par une séquence du moniteur.
- un pointeur permet d'accéder à l'information d'événement de l'opération bloquante. Dans le cas d'un accès au terminal, il s'agit de la variable d'événement gérée par la méthode d'accès TAM (ECB). Le cas de l'attente de synchronisation sera décrit ultérieurement.

Algorithme de choix de la tâche à activer [VIII]

Parmi les tâches candidates, la séquence de commutation doit choisir la tâche à activer. Elle choisit toujours d'activer celle qui n'a pas eu la possibilité de recevoir le contrôle de l'unité centrale depuis le plus longtemps : en effet, la liste des tâches est consultée cycliquement ; le moniteur active la première tâche candidate parmi celles situées après la dernière tâche activée dans la liste.
Pour faciliter cette consultation cyclique de la liste des tâches, les tables TCT sont chaînées en liste circulaire :



Remarque : si aucune tâche n'est candidate à l'activation, l'application entière est mise en attente d'un de événements bloquants.

L'algorithme de choix de la tâche à activer peut être formal comme suit :

```
Début
Tâche courante : = dernière tâche active ;
Fin de liste : = dernière tâche active ;
Candidat trouvé : = faux ;
Tant que Candidat trouvé = faux faire
     Début
     Tâche courante : = suivant (tâche courante) ;
     Si Tâche courante directement activable
        ou Tâche courante non directement activable
        et évènement bloquant arrivé
     Alors Candidat trouvé : = vrai
     Sinon Si tâche courante = fin de liste
           Alors Attendre un évènement
           Fsi
     Fsi
     Fin ;
Activer tâche courante ;
Fin ;
```

Utilisation du niveau commutation (répartiteur) par le syst

Lorsqu'une séquence système demande une commutation, elle ef tue un appel au niveau traitant les commutations.

Avant l'appel de la commutation, la séquence travaille pour compte d'un usager. Lorsqu'elle reprend le contrôle à l'inst tion suivant l'appel du niveau commutation, le contexte de la tâche a été restauré, la séquence continue le travail pour le compte du même usager. Toute l'opération de commutation, et toutes les activations des autres tâches ont été totalement transparentes pour la séquence, car tout a été traité par le niveau commutation. Les séquences systèmes correspondant au traitement de la tâche usager se présentent donc comme une séquence de programme continue, les "points interruptibles" étant traités par de simples appels au niveau : commutation.

Remarque sur le contexte des tâches

Lorsqu'une commutation de tâches est demandée, les registres de l'ordinateur ne doivent pas être significatifs (sauf le registre d'index de réentrance et le registre utilisé par l'instruction d'appel). En effet, aucune sauvegarde des registres n'est effectuée.

Si les commutations étaient forcées par un système de Time Slicing, les interruptions horloge pourraient arriver alors que les registres seraient significatifs. Une sauvegarde systématique serait alors nécessaire à chaque commutation.

D'où un gain de temps de calcul sur les sauvegardes de contexte.

Gestion de la mémoire du logiciel d'application

Si une commutation est demandée alors qu'aucun service d'application n'est en cours pour le compte de la tâche à désactiver, la mémoire libre réservée au logiciel d'application n'est pas significative pour la tâche. Si au contraire, un service est en cours, des swappings de sauvegarde doivent être effectués. Il en est de même pour les swappings de restauration à la réactivation.

La séquence de commutation ne prenant pas en charge ces opérations, les swappings de sauvegarde doivent être effectués immédiatement avant l'appel du niveau de commutation, les swappings de restauration immédiatement après.

Toutes ces opérations pour le compte d'une même tâche sont prises en charge par un niveau qui sera appelé à la place du niveau de commutation simple quand un service d'application est en cours d'exécution.

Ce niveau effectue donc les opérations suivantes dans l'ordre :

- sauvegarde de la zone des variables du service en cours sur l'espace réservé à la tâche dans le fichier de swap-

ping.

(aucun swapping de sauvegarde n'est nécessaire pour l texte du service d'application qui n'est jamais modif

- commutation des tâches proprement dite, par appel du veau : commutation simple (répartiteur).
- restauration de la zone des variables du service en c à partir de la sauvegarde sur fichier externe effectu à la désactivation.

Si le texte du service n'est pas déjà présent en mémo chargement du texte de ce service à partir du fichier services.

Description des informations nécessaires

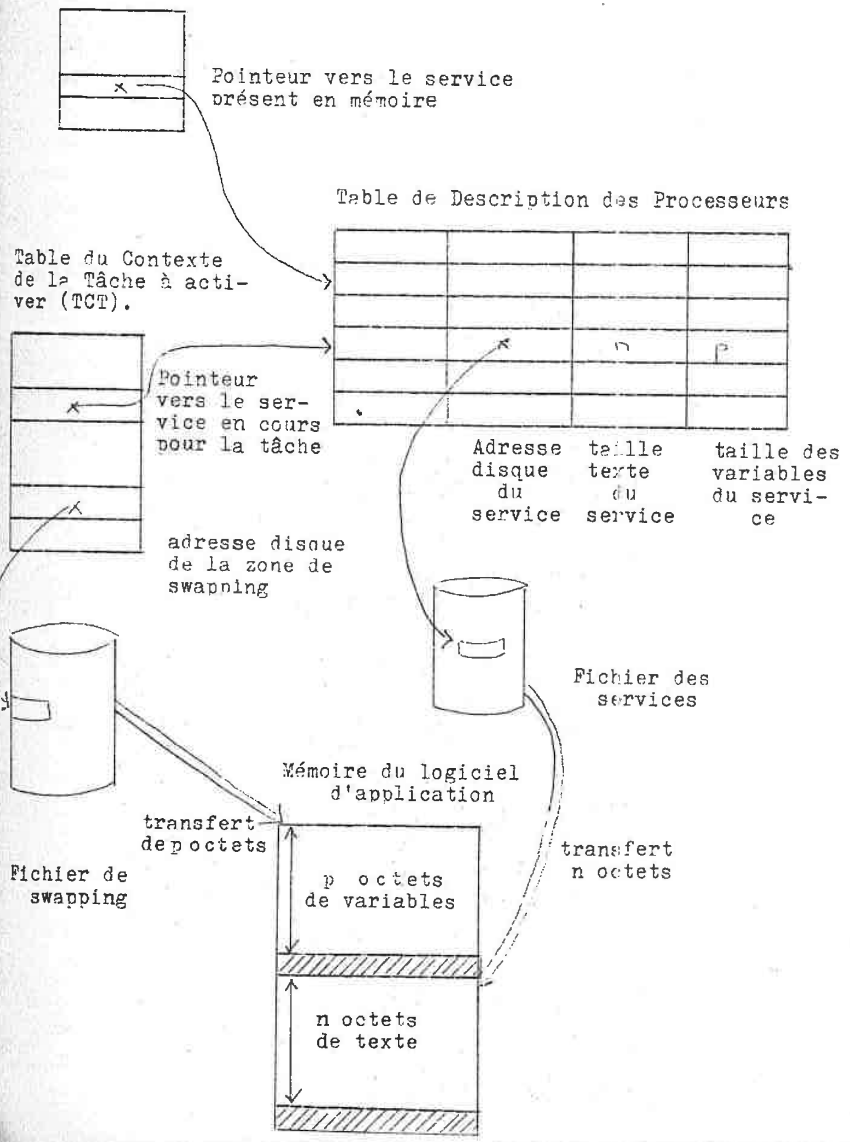
Pour effectuer les opérations de restauration à l'activatio la séquence de gestion mémoire doit disposer des informatio suivantes :

- identification du service présent en mémoire.
- identification du service en cours pour la tâche.
- adresse dans le fichier des services, du texte du serv en cours pour la tâche.
- taille du texte du service en cours pour la tâche.
- adresse dans le fichier de swapping, de la zone réser à la tâche.
- taille de la zone des variables du service en cours p la tâche.

Ces informations sont suffisantes pour les opérations de sau garde à la désactivation. Pour obtenir ces informations, le niteur SAFRAN dispose :

- d'une Table de Description des Processeurs (TDP) décri vant les services d'application et contenant (entre au pour chaque service :
  l'adresse du texte correspondant dans le fichier des vices.
  la taille du texte correspondant.
  la taille de la zone des variables correspondante.
- des Tables des Contextes des Tâches (TCT) contenant po chaque tâche :
  un pointeur vers l'entrée dans la table TDP décrivant service en cours.
  l'adresse dans le fichier de swapping, de la zone rése vée.
- d'une table de renseignements généraux du moniteur (TRG) contenant : un pointeur vers l'entrée dans la table TDP décrivant le service en mémoire.

Le schéma suivant met en évidence la distribution de ces informations et leur utilisation pour les swappings de restauration (les swappings de sauvegarde, plus simples, s'en déduisent aisément).



Accès aux terminaux

Traitement par niveaux

Les opérations de lecture ou d'écriture sur le terminal asso
à la tâche sont des opérations élémentaires appelées à parti
de plusieurs points du moniteur. La lecture au terminal est
ployée, par exemple :

- pour obtenir une demande de service de la part d'un u
  (état exécutif).
- pour satisfaire une demande de saisie d'information la
  par le logiciel d'application etc...

Il est donc naturel de faire gérer ces opérations par des ni
spécialisés. Les caractéristiques propres à chaque accès peu
être passées en paramètres au moment de l'appel du niveau (d
les registres, par exemple).
Ces opérations provoquent toujours le passage de la tâche da
des états d'attente logique, pendant l'exécution de l'opérat
Elles doivent donc s'accompagner de commutations de tâches.
Par ailleurs, les niveaux d'accès au terminal doivent être a
lés alors qu'un service d'application est actif ou non. Dans
tains cas, la commutation des tâches doit donc s'accompagner
la gestion de la mémoire libre.
Pour cela, quatre niveaux sont utilisés par le moniteur SAFR

- lecture au terminal appelant une commutation sans gest
  de mémoire.
- lecture au terminal appelant une commutation avec gest
  de mémoire.
- écriture au terminal appelant une commutation sans ge
  de mémoire.
- écriture au terminal appelant une commutation avec ges
  de mémoire.

Informations nécessaires

- L'accès au terminal est réalisé par un appel-moniteur SIRI
  de la méthode d'accès TAM. Pour permettre l'accès au termi
  associé à la tâche, le moniteur SAFRAN doit fournir à SIRI
  l'index de ce terminal, dans la liste des terminaux gérés
  TAM pour le compte de l'application.
- Des opérations d'accès aux terminaux peuvent être en cours
  simultanément pour le compte de plusieurs tâches. Chaque t
  doit donc posséder son propre tampon afin d'éviter les mé
  ges d'informations en cas d'accès simultanés.
  Pour être accessible, l'adresse de ces tampons doit figurer
  dans la table TCT de la tâche.

N.B. : Tous les accès aux terminaux n'utilisent pas ces tampons.
L'écriture de messages fixes, par exemple, peut référencer directement le message comme tampon.

Traitements effectués par les niveaux.

- Récupération des paramètres décrivant l'opération.
- Lancement de l'opération d'entrée-sortie (appel-moniteur SIRIS 8-TAM).
- Affectation de la variable logique de la table TCT indiquant une candidature conditionnelle à l'activation.
- Introduction du pointeur d'événement fourni par TAM et associé à l'opération, dans la table TCT, pour permettre la consultation de cet événement par la séquence de commutation des tâches. La tâche pourra ainsi être réactivée à l'achèvement de l'opération physique d'accès.
- Demande de commutation par appel au niveau : commutation simple ou au niveau : commutation avec gestion de la mémoire du logiciel d'application.
- Achèvement de l'opération d'entrée-sortie (appel-moniteur SIRIS 8-TAM) car la tâche a alors été réactivée, l'accès est terminé.
- Passage du code d'achèvement de l'opération en paramètre, dans un registre (code fourni par TAM).

FONCTIONNEMENT DU MONITEUR SAFRAN

(tâche usager)

Gestion des demandes de connexions

Prise en compte des terminaux non opérationnels

Lorsqu'une application SAFRAN est mise en exploitation réelle, elle est lancée sur demande de l'opérateur central.
Au moment du lancement de l'application, tous les terminaux ne sont généralement pas opérationnels, et l'opérateur central ne doit pas avoir à s'en préoccuper. Pourtant, l'application doit être en mesure de recevoir les demandes de connexions en provenance des terminaux, au moment même où ces derniers les demandent. De même un usager ayant achevé son travail peut vouloir rompre la connexion physique avec le central, pour une utilisation locale de son matériel, par exemple. Ceci ne doit pas nuire au fonctionnement du système. L'usager doit même pouvoir redemander une connexion ultérieurement.
Cette prise en compte de terminaux non opérationnels ne peut être effectuée que par usage de l'opération de surveillance offerte par la méthode d'accès TAM.

Réception des demandes de connexions

Lorsqu'un terminal n'est pas connecté, la tâche correspondante est dans l'état : inexistant. Ceci correspond à un état d'attente logique qui s'achève à la réception d'une demande de connexion. En pratique, une tâche passe dans l'état inexistant après avoir opéré une mise en surveillance du terminal associé.
La tâche inexistante est donc en fait tout simplement en attente de l'évènement associé à la surveillance. La tâche est donc inactive. Elle deviendra candidate à l'activation à la réception d'un caractère "Attention" frappé au terminal, et achevant l'opération de surveillance. La frappe d'un tel caractère est

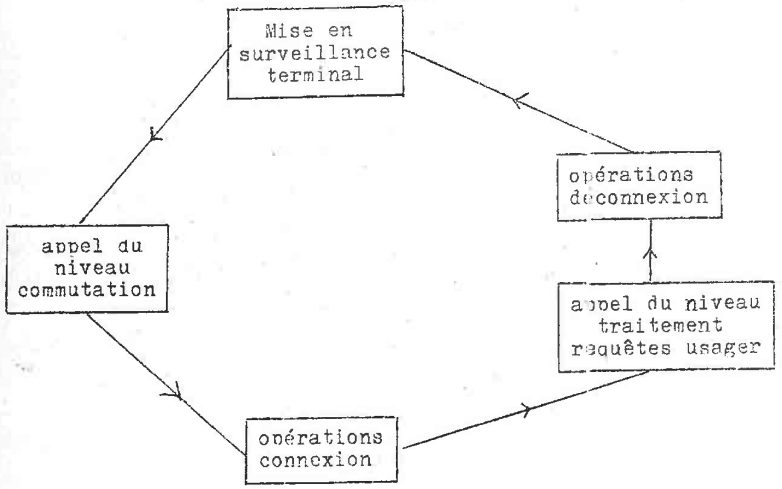
donc considérée comme une demande de connexion.

Traitement des demandes de connexions

Les demandes de connexions des usagers sont prises en charge par un niveau qui exécute toutes les opérations associées aux connexions et aux déconnexions :

- au départ, la tâche est en attente d'un évènement de surveillance. Elle est donc inactive.
- la tâche est activée à l'occasion d'une commutation de tâche après achèvement de la surveillance.
- la tâche effectue alors les opérations de connexion, si aucun arrêt en douceur n'a été demandé :
  envoi au terminal des messages de début de session (appel du niveau). L'indicatif SAFRAN du terminal lui est alors envoyé
  mise à jour du compte de terminaux connectés.
  envoi d'un message à l'opérateur central, l'informant de l'état de la charge.
- la tâche est prête à passer à l'état exécutif et à traiter des demandes de services. Toutes ces opérations sont prises en charge par un niveau qui est appelé ici (traitement des requêtes usager).
- le niveau : traitement des requêtes usager reçoit les demandes de déconnexions ; il rend donc la main pour le traitement des déconnexions.
- la tâche effectue alors les opérations de déconnexion :
  envoi au terminal des messages de fin de session (appel du niveau).
  mise à jour du compte de terminaux connectés.
  envoi d'un message à l'opérateur central l'informant de l'état de la charge.
- une opération de surveillance est lancée sur le terminal associé.
- la tâche demande sa désactivation par appel du niveau : commutation des tâches (sans gestion de la mémoire libre).
  Elle est alors prête à recevoir une nouvelle demande de connexion.

Le schéma suivant visualise la logique de ce niveau :



Initialisation

Au lancement d'une application, toutes les tâches sont inexistantes. La séquence d'initialisation doit donc lancer une opération de surveillance sur tous les terminaux et initialiser toutes les tables TCT comme avant une demande de commutation. L'application peut alors être mise en attente comme à l'issue d'une demande de commutation quand aucune tâche n'est candidate. Chaque tâche peut alors être activée si l'adresse de départ du traitement des connexions a été au préalable placée dans la pile de chacune : en effet, lorsqu'une tâche sera candidate à l'activation, le module de commutation des tâches tentera de l'activer à l'adresse contenue au sommet de la pile.
Le niveau : traitement des connexions et déconnexions est donc le plus bas dans l'ordre des appels.

N.B. L'initialisation de la pile ne doit être faite qu'au départ, car en fin de session, cette adresse est automatiquement empilée à l'appel du module de commutation des tâches, en raison de la structure en boucle du niveau.

Traitement des requêtes-usager

Types de services offerts

Deux types de services sont offerts à l'usager par une applica-

sion SAFRAN :

- les services standard et les services ajoutés
  (appelés aussi : commandes immédiates, par analogie avec SIRIS 8 TS). Ce sont des séquences de programme faisant logiquement partie du texte du moniteur, appliquant donc ses conventions, et exécutant des fonctions précises pour le compte de l'usager. Le texte de ces services est toujours résident. La version actuelle du moniteur SAFRAN offre un seul service standard (émission d'un message vers un autre terminal). D'autres services peuvent être ajoutés sans modification du moniteur. La marche à suivre pour les insérer sera décrite ultérieurement.
- les services d'application
  (appelés aussi processeurs, par analogie avec SIRIS 8 TS). Ce sont des séquences de programme écrites exclusivement pour une application donnée et respectant les règles imposées au logiciel d'application. Elles ne sont pas résidentes, mais stockées sur un fichier de services. Elles sont chargées en zone libre pour exécution. Ce sont normalement les seuls programmes à concevoir pour construire une application SAFRAN. Les services ajoutés ne doivent être employés qu'en cas de besoin particulier (impossibilité de traiter en séquences d'application, nécessité de résidence etc...).

A chacun de ces services (standard, ajoutés, d'application) correspond un nom qui le désigne de façon unique dans l'application, et permet aux usagers d'en demander l'exécution.

Obtention des requêtes usager

Lorsque le niveau : traitement des requêtes usager reçoit le contrôle, la tâche usager a exécuté les fonctions de connexion. Elle est prête à exécuter des services. Pour obtenir des requêtes de l'usager, elle lance une lecture au terminal et passe ainsi à l'état exécutif. L'usager peut alors demander l'exécution d'un service par simple mention de son nom.
Les services standard, les services ajoutés, les services d'application sont demandés de la même façon par l'usager du terminal. Le type du service lui est totalement transparent.
Pour permettre le lancement du service demandé, le moniteur doit disposer de la liste des noms des services disponibles dans l'application. Ces noms sont groupés dans deux tables :

- la Table de Description des Commandes immédiates (TDC).
  Elle contient la liste des services standard et ajoutés.

- la Table de Description des Processeurs (TDP).
  Elle contient la liste des services d'application.

Si le nom demandé ne figure dans aucune de ces tables, un message d'erreur est envoyé à l'usager et la tâche repasse dans l'état exécutif.

Exécution des services standard ou ajoutés

Les services standard (un seul dans la version actuelle) et les services ajoutés sont traités de la même façon : la table TDC fait correspondre au nom de chacun l'adresse de la séquence correspondante.
L'exécution du service est lancée sans changement de niveau ; son texte fait donc partie du niveau : traitement des requêtes usager. Il peut appeler d'autres niveaux :
Ecriture, lecture au terminal, commutation (sans gestion mémoire, car aucun service d'application n'est en cours).
En fin d'exécution, la tâche repasse en état exécutif.

Demandes de déconnexion

Les anomalies généralement détectées à l'opération de lecture (état exécutif) sont les suivantes :

- rupture de ligne
- pas de réponse à la demande au bout d'un temps limité
- caractère spécial frappé au terminal

Les codes d'achèvement correspondants fournis par la méthode d'accès TAM au moment du test d'entrée-sortie, sont restitués au moniteur SAFRAN au retour du niveau d'accès au terminal. Ils peuvent donc être testés.

- Une rupture de ligne impose une déconnexion forcée du terminal.
- Un dépassement du délai de réception est considéré comme un abandon du terminal par l'usager. Une déconnexion est forcée.
- L'émission d'un caractère spécial (Attention, par exemple) est considérée comme une demande de déconnexion volontaire.

Dans tous les cas d'anomalie à la lecture, le contrôle est rendu au niveau de traitement des déconnexions.

Exécution des services d'application

Insertion logique d'une séquence d'application dans le moniteur SAFRAN

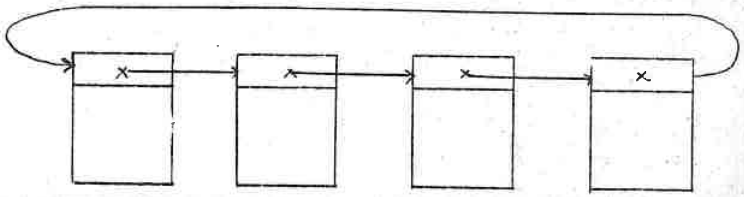
Lorsqu'une demande d'exécution d'un service d'application a été reçue par le niveau : traitement des requêtes usager, le texte de la séquence correspondante peut être chargé dans la mémoire libre réservée au logiciel d'application. Tous les renseignemen nécessaires pour cette opération sont contenus dans la Table de Description des Processeurs (TDP). Cette table fait correspondre à chaque nom de service, les paramètres de ce service :

- adresse de la séquence correspondante dans le fichier des services.
- taille du texte de la séquence correspondante.
- taille de la zone des variables utilisée.

Après chargement du service, l'exécution peut être lancée.
Par hypothèse, l'exécution de ce service ne se déroule pas de bout en bout, mais est périodiquement interrompue par des appel au moniteur SAFRAN provoquant la commutation des tâches, jusqu'à l'appel final provoquant le retour à l'état exécutif.
Pendant l'exécution du service, les commutations sont traitées par des appels au niveau : commutation avec gestion mémoire, la zone réservée au logiciel d'application étant alors significative pour la tâche.
Les informations nécessaires au fonctionnement de ce niveau devront donc avoir été mises en place au préalable, c'est à dire avant le lancement de la séquence d'application. Elles se réduisent au pointeur, dans la table TCT, vers l'entrée dans la table TDP correspondant au service en cours d'exécution pour la tâche (voir commutation).

Remarque sur l'ordre d'activation des tâches

Par hypothèse, entre deux commutations, les tâches exécutent une séquence de calcul courte par rapport au temps d'une Entrée-Sortie au terminal. Par ailleurs, les tables des contextes des tâches sont examinées cycliquement par le module de commutation



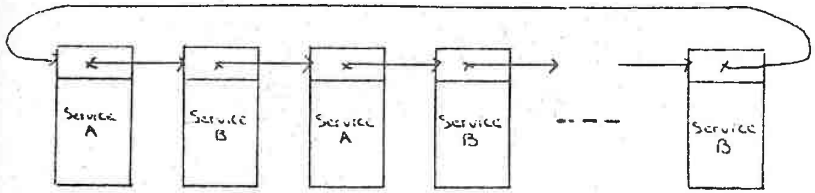
- Si les tâches s'interrompent pour des accès aux terminaux et que peu d'usagers sont connectés à l'application, l'unité centrale de l'ordinateur est peu employée. Elle est donc en principe disponible dès qu'une tâche devient candidate, l'application étant généralement en attente.
  Les tâches sont donc activées dans un ordre quelconque, fonction de la durée des Entrées-Sorties réelles, et non de l'ordre de consultation.
- Si les tâches s'interrompent pour des accès aux terminaux et que le système est très chargé, l'unité centrale de l'ordinateur est très employée. Le temps de parcours de la liste des tâches est long. Lorsqu'une tâche est consultée, elle ne l'a pas été depuis longtemps, elle est donc probablement candidate.
  Les tâches sont alors activées dans l'ordre de la liste.
- Si les tâches s'interrompent pour des commutations volontaires, il en est de même, même s'il y a peu d'usagers actifs : cet état correspond à une charge importante en unité centrale.

En résumé, lorsque le système est peu chargé les tâches sont activées dans un ordre aléatoire. Lorsque le système est chargé, elles sont activées dans l'ordre de la liste.

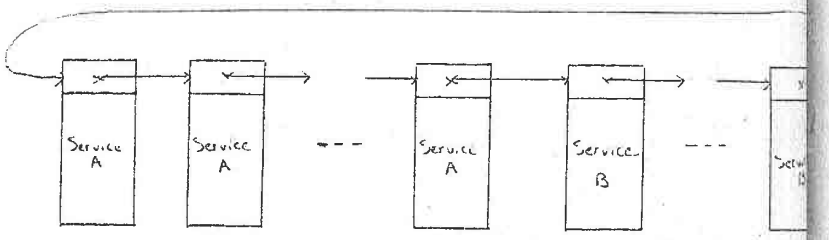
Remarque sur la disposition de la liste des tâches

A un instant donné, plusieurs tâches peuvent exécuter le même service d'application, alors que d'autres en exécutent un autre. Plusieurs cas de disposition de ces tâches sont possibles dans la liste parcourue par la séquence de commutation.
Voici deux cas de disposition différents pour le même travail : sur n tâches, n/2 exécutent le service A, n/2 le service B :

1er cas :



2ème cas :

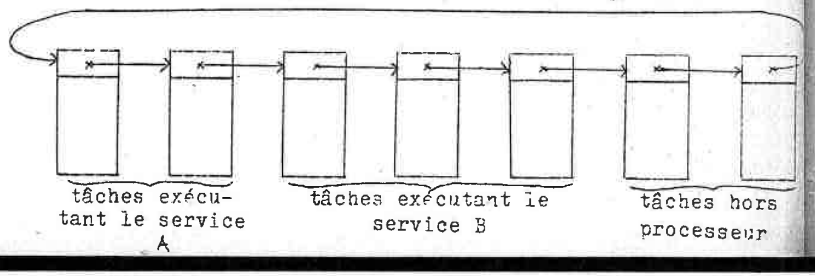


- Si le système n'est pas chargé, les tâches sont activées dans un ordre aléatoire. Les deux dispositions sont équivalentes.
- Si le système est chargé, les tâches sont activées dans l'ordre de la liste. Pendant le temps d'exécution d'une boucle :
  dans le premier cas, le texte du service A est chargé n/2 fois, le texte du service B est chargé n/2 fois, soit n chargements.
  Dans le deuxième cas, le texte du service A est chargé 1 fois, le texte du service B est chargé 1 fois, soit 2 chargements.
  (les zones de variables sont par contre swappées systématiquement).

Le système étant chargé, tous les gains sont appréciables, le deuxième cas semble donc plus favorable que le premier.

Organisation réelle de la liste des tables TCT

Afin de n'imposer qu'un minimum de swappings, la liste des TCT doit être organisée de façon telle, qu'à un instant donné, toutes les tâches utilisant le même service d'application se trouvent groupées dans une même sous-liste. Les tâches n'exécutant pas de service d'application (dites tâches hors processeur) sont elles aussi, groupées en une sous-liste.



Pour respecter en permanence cette organisation, la liste doit être modifiée toutes les fois qu'une tâche change d'état vis à vis des services, à savoir au lancement et à la fin d'exécution de chaque service d'application, pour chaque tâche.
Le lancement d'un service d'application s'accompagne donc des opérations suivantes :

- déchaîner la tâche de la sous-liste des tâches hors processeur.
- si d'autres tâches utilisent le même service, rechaîner la tâche en fin de sous-liste des tâches utilisant le service choisi.
- si aucune tâche n'utilise déjà ce service, créer une sous-liste entre deux sous-listes existantes (en pratique, la tâche est chaînée après la sous-liste des tâches hors processeur).

La fin d'exécution d'un service d'application s'accompagne elle-même des modifications suivantes de la liste des tâches :

- déchaîner la tâche de la sous-liste des tâches utilisant le service achevé.
- si d'autres tâches sont hors processeur, rechaîner la tâche en fin de sous-liste des tâches hors processeur.
- si aucune tâche n'est déjà hors processeur, créer la sous-liste entre deux sous-listes existantes.

N.B. La sous-liste des tâches hors processeur n'est pas indispensable : ces tâches n'ayant aucune action sur les swappings, elles peuvent être réparties sans perturbations entre les tâches exécutant un service. Cette organisation n'a été choisie que parce que les adjonctions et modifications apportées au moniteur sont facilitées par une organisation "propre" du programme.

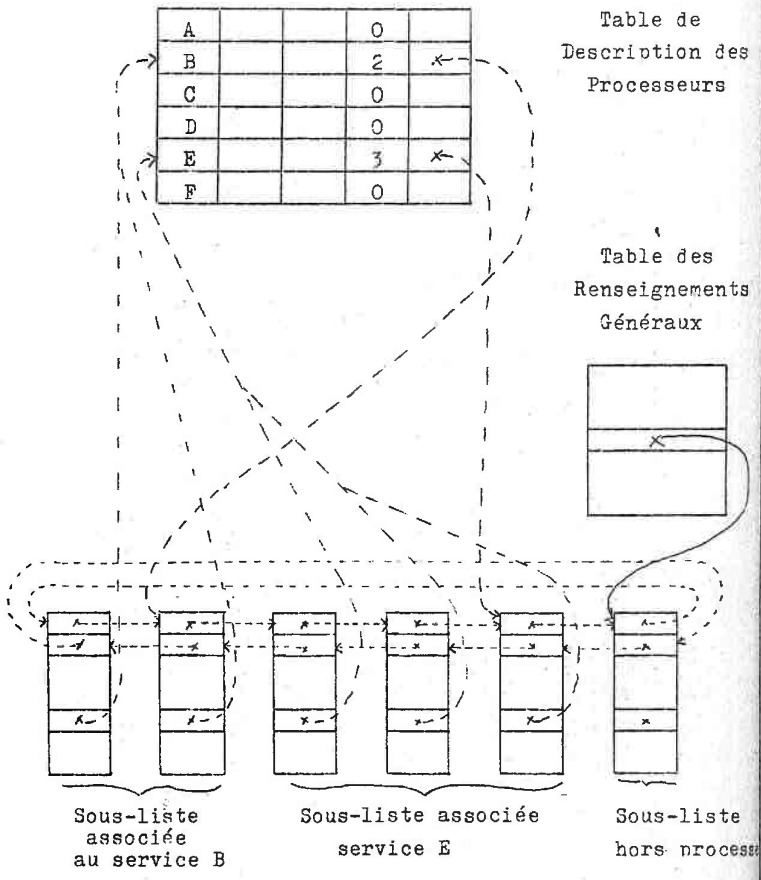
Informations nécessaires

- Les modifications de la liste des tables TCT imposent un double chaînage (amont et aval).
- A chaque service doivent être associés :
  un compteur des tâches utilisant le service,
  lorsque ce compteur n'est pas nul, un pointeur vers la dernière table TCT de la sous-liste associée au service.
  Ces informations sont contenues dans la table de description des processeurs (TDP).
- Pour l'ensemble de l'application, il doit exister un pointeur vers la fin de la sous-liste des tâches hors proces-

seur. (lorsqu'il n'y a plus de tâches hors processeur, ce pointeur désigne une fin de sous-liste quelconque). Ce pointeur est contenu dans la Table des Renseignements Généraux du système (TRG).

- Pour chaque tâche il doit exister :
  une variable logique indiquant si la tâche est hors processeur,
  si non, un pointeur vers l'entrée dans la table TDP correspondant au service actif.
  Ces informations sont contenues dans les Tables des Contextes de Tâches (TCT).

Le schéma suivant visualise l'organisation des informations :



Algorithmes de modification

Les modifications de listes peuvent être formalisées par les algorithmes suivants :

- au lancement :

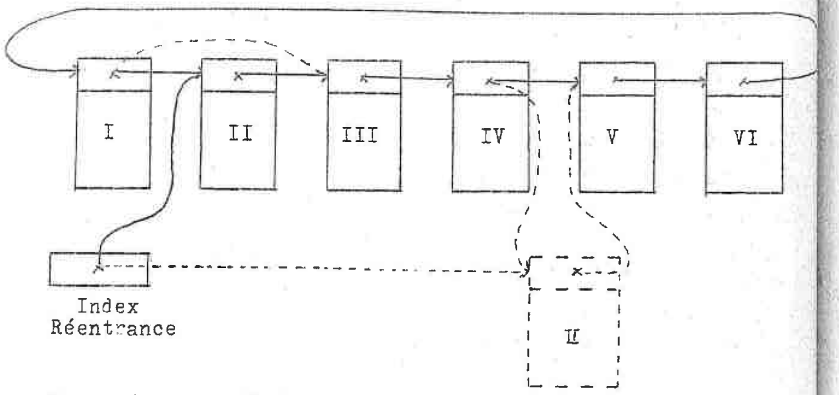
  Début
  Si tâche en cours = fin de sous-liste hors processeur
  Alors fin de sous-liste hors processeur : = précédent (tâche en cours) ;
  Si Sous-liste associée au service non vide
  Alors position : = fin de sous-liste associée au service
  Sinon position : = fin de sous-liste hors processeur ;
  Chaîner tâche en cours dernière position ;
  fin de sous-liste associée au service : = tâche en cours ;
  compte des tâches associées au service : = compte des tâches associées au service +1 ;
  Si précédent (tâche en cours) = fin de sous-liste hors processeur et précédent (tâche en cours) hors processeur
  Alors fin de sous-liste hors processeur : = tâche en cours ;
  Fin ;

- à la fin d'exécution :

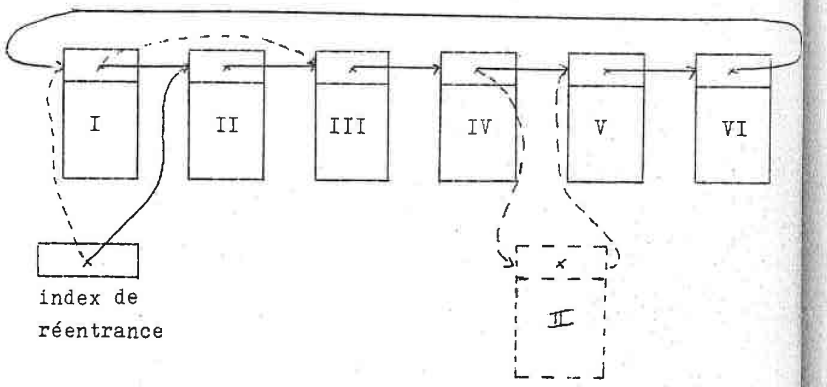
  Début
  Si tâche en cours = fin de sous-liste hors processeur
  Alors fin de sous-liste hors processeur : = précédent (tâche en cours) ;
  Si tâche en cours = fin de sous-liste associée au service
  Alors fin de sous-liste associée au service : = précédent (tâche en cours).
  compte des tâches associées au service : = compte des tâches associées au service -1 ;
  chaîner tâche en cours derrière fin de sous-liste hors processeur ;
  fin de sous-liste hors processeur : = tâche en cours ;
  Fin ;

Incidence des modifications sur l'ordre des activations

Si une modification de liste est effectuée immédiatement avant le lancement, un certain nombre de tâches en aval de l'ancienne position vont perdre un tour de consultation. Il en est de même en fin d'exécution.



Le schéma pointillé indique les modifications subies par la liste des tables TCT.
Après désactivation de la tâche II, la consultation séquentielle de la liste reprendra à la tâche V. Même si les tâches III et IV étaient candidates, elles ne seront activées qu'au passage suivant de la séquence de commutation sur la liste.
Une telle situation peut être évitée si une commutation est forcée après modification de liste, avec départ de la recherche de la tâche candidate sur la tâche précédant l'ancienne position de la table déplacée.
La tâche déplacée sera alors réactivée à son tour dans la nouvelle liste, mais aucune tâche ne perdra de tour d'activation.



Réception des signaux asynchrones de l'usager

Pendant l'exécution d'un service d'application, l'usager doit pouvoir transmettre à ce service des signaux, même lorsqu'aucune opération de lecture n'est lancée au terminal. Ceci est rendu possible par l'opération de surveillance parallèle offerte par la méthode d'accès TAM. Une surveillance peut être lancée sur un terminal quel que soit son état, qu'il soit opérationnel ou non. D'autres opérations peuvent être lancées après sur le même terminal, et se terminer avant. Cette surveillance ne s'achève que lors de la réception d'un caractère Attention en provenance du terminal.
Pour détecter les demandes asynchrones de l'usager, une surveillance doit donc être lancée sur le terminal en même temps que le service d'application, et purgée en fin d'exécution du service.
L'événement associé à cette surveillance fait partie du contexte de la tâche : son adresse, fournie par TAM au lancement de la surveillance doit donc être rangée dans la table TCT associée à la tache. Cet événement est testé périodiquement, à chaque appel au système lancé par le service d'application. Quand un caractère attention est émis par l'usager, un code special est memorisé dans la table TCT de la tâche. Ce code est accessible au service, qui doit le tester périodiquement (voir traitement des requêtes-processeurs).

Passage d'information entre services successifs

Lorsque deux services d'application sont successivement appelés par le même usager, des informations peuvent être fournies par le premier au deuxième. Ces informations sont contenues dans
- les registres de l'ordinateur
- la zone des variables

Pour cela, les valeurs des registres doivent être sauvegardées à la fin d'exécution d'un service d'application et restaurées au lancement du suivant. Une zone de 16 mots est réservée pour cet usage à chaque tâche. L'adresse de cette zone est contenue dans la Table du Contexte de la Tâche (TCT).
De même la zone des variables doit être sauvegardée à la fin d'exécution d'un service, et restaurée au lancement du suivant. Ces copies se font sur le fichier de swapping.
Remarques :
- si la zone des variables du 2° service est plus courte que celle du 1er, les informations contenues en fin de zone du 1er service sont perdues.
- si la zone des variables du 2° service est plus longue que celle du 1er, seul le début de la zone

du 2° service est significatif.

- s'il désire utiliser ces services, le concepteur de l'application doit s'assurer lui-même de la compatibilité de ces zones de variables, qui doivent alors être gérées comme des communs.

Lancement et fin des services d'application

En résumé, les opérations effectuées au lancement d'un service d'application sont les suivantes :

- modification de la liste des tables TCT.
- commutation, pour respecter l'ordre des activations.
- à la réactivation, lancement de la surveillance du terminal
- restauration de la mémoire réservée au logiciel d'application (texte et variables).
- restauration des registres.
- lancement du service.

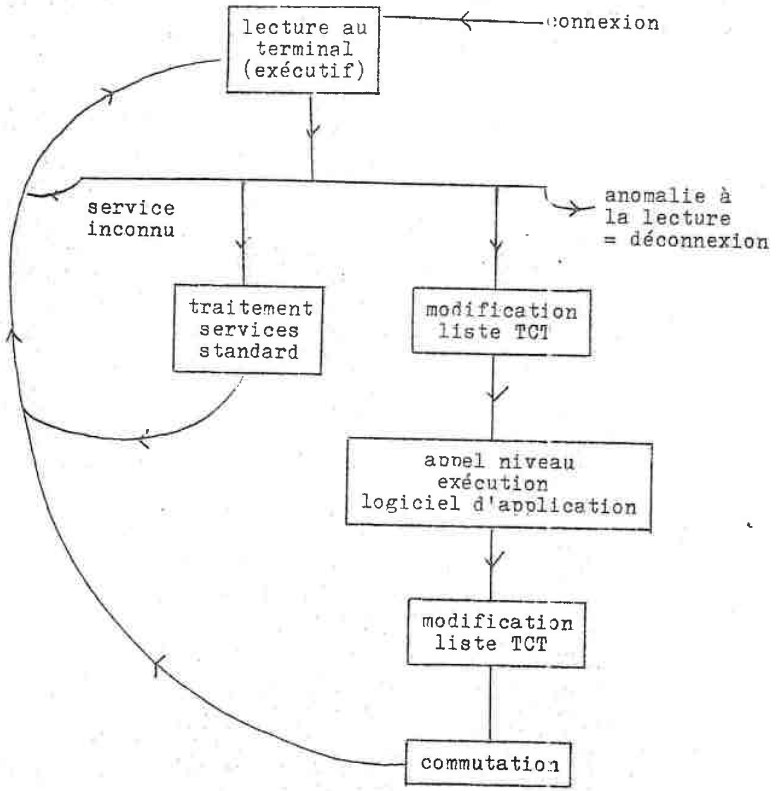
La fin d'exécution d'un service d'application est détectée par un appel-système. La séquence correspondante, qui sera décrite ultérieurement rend le contrôle aux séquences traitant l'exécution des services. Les opérations suivantes sont alors effectuées :

- la sauvegarde des registres a déjà été faite (traitement des requêtes processeurs).
- sauvegarde de la zone des variables de la séquence d'application.
- purge de la surveillance du terminal.
- modification de la liste des tables TCT.
- commutation pour respecter l'ordre des activations.
- à la réactivation, passage à l'état exécutif.

En fait, ces opérations sont exécutées sur deux niveaux :

- il est naturel que certaines soient exécutées sur le niveau : traitement des requêtes usager.
- la séquence d'application doit logiquement faire l'objet d'un niveau isolé.

En pratique, toutes les opérations correspondant à la gestion de la liste des tables TCT sont traitées sur le niveau : traitement des requêtes usager. Les opérations d'interface avec le logiciel d'application sont traitées sur le niveau : exécution du logiciel d'application (surveillance du terminal, sauvegarde restaurations). La logique générale du niveau : traitement des requêtes usager peut être représentée par le schéma suivant :



Traitement des requêtes-processeurs

Mise en oeuvre des déroutements programmés sous SIRIS 8 [X]

L'IRIS 80 offre quatre instructions permettant la génération de déroutements :

CAL1 CAL2 CAL3 CAL4

Dans les programmes exécutés sous contrôle du moniteur SIRIS 8, l'instruction CAL1 est réservée à l'appel des services du système (appels-moniteur). Les instructions CAL2, CAL3, CAL4 ne sont pas utilisées par SIRIS 8, mais leur emploi dans un programme standard conduit généralement à un abandon prématuré du travail.

Un service offert par le moniteur SIRIS 8 permet d'éviter l'avortement du programme sur emploi de ces instructions. Ce service est appelé par la procédure M:TRAP.

Lorsque ce service a été appelé en début de programme, toute exécution d'une instruction CAL2, CAL3, CAL4 a l'effet suivant :

- l'exécution du programme est interrompue après l'instruction.
- l'exécution est relancée à une adresse qui a été fournie à SIRIS 8 à l'appel du service M:TRAP en début de programme. La séquence dont l'exécution commence est appelée séquence d'exception.
- à l'entrée dans la séquence d'exception, le registre 5 contient l'adresse d'un mot contenant l'adresse de l'instruction CAL responsable de l'interruption du programme.
- une telle séquence doit obligatoirement se terminer par un appel-moniteur spécial qui, appelé par la procédure M:RETURN a l'effet suivant :
  les registres 5, 6, 7, 8 sont rechargés dans l'état où ils se trouvaient avant l'interruption du programme, donc avant l'exécution de l'instruction CAL.
  l'exécution de la séquence d'exception est arrêtée, et le programme est normalement relancé à l'adresse fournie à SIRIS 8 lors de l'appel du service M:RETURN.
  (si les registres 5, 6, 7, 8 sont les seuls modifiés par la séquence d'exception, tous les registres se retrouvent après l'exécution de M:RETURN, dans le même état qu'avant l'exécution de l'instruction CAL.)

Utilisation des déroutements programmés

Les services de déroutements programmés permettent une forme d'appel de sous-programmes ; en effet :

- l'instruction CAL provoque un branchement.
- le registre 5 permet de récupérer l'adresse de l'instruction d'appel, donc d'éventuels paramètres situés dans la zone "adresse" de l'instruction CAL (inutilisée par l'instruction) ou après l'instruction CAL.
- l'adresse de l'instruction étant connue, l'adresse de retour peut être calculée en fonction du nombre de paramètres éventuels.

La seule différence logique avec l'appel standard des sous-programmes (instruction BAL) est que l'adresse de traitement de la fonction appelée n'est pas contenue dans l'instruction.

Appel des fonctions SAFRAN par un service d'application

Par hypothèse, l'exécution d'un service d'application n'est jamais continue : les séquences doivent s'interrompre régulièrement pour appeler les services offerts au logiciel d'application (au minimum sont ainsi appelées les fonctions de commutation et d'accès aux terminaux). Afin de garantir l'indépendance du texte du moniteur et du texte des services d'application, l'adresse de la séquence de traitement des fonctions appelées ne doit pas figurer dans l'instruction d'appel. Pour cela, les appels sont effectués par déroutements programmés. (Le service SIRIS 8 M:TRAP doit donc être appelé au lancement de l'application.)

La même instruction CAL2 est employée pour l'appel de toutes les fonctions. Un code dans la zone adresse de cette instruction précise le service demandé (code d'appel-système).

Lorsque les paramètres doivent être fournis au service (par exemple adresse du tampon dans le cas d'un accès au terminal), ces paramètres sont groupés dans une table placée immédiatement derrière l'instruction CAL. Ces tables, dites Tables d'Interface Système (TIS) ont une structure différente pour chaque fonction. La description des appels de toutes les fonctions standard SAFRAN est disponible en annexe A.

Réception des demandes de fonctions

- Le traitement des fonctions SAFRAN nécessite généralement l'emploi des registres (en particulier, en cas de commutation !).
  Or la sauvegarde des registres du service d'application avant l'appel et leur restauration au retour est une opération pénible qu'il convient d'éviter au logiciel d'application. Les opérations de sauvegarde doivent donc être prises en charge par le moniteur à la réception des demandes de fonctions.
- Le traitement des fonctions nécessite aussi l'accès aux variables propres à la tâche en cours. Celles-ci ne peuvent être obtenues que par la connaissance de l'adresse de la table TCT associée à la tâche (valeur de l'index de réentrance). Pour cela, une copie de la valeur de cet index est tenue à jour, à chaque commutation de tâches, dans la Table des Renseignements Généraux (TRG), par le niveau : commutation.

Les opérations effectuées à la réception des demandes de fonctions sont les suivantes :

- sauvegarde (dans la table TRC) de l'adresse de l'instruction CAL origine de l'appel ; en effet cette adresse est disponible à la réception de la demande, car cette réception se fait en séquence d'exception, mais elle ne le serait plus après lancement de la fonction, qui se fait après retour de séquence d'exception.
- retour de séquence d'exception, afin de restaurer les registres 5, 6, 7, 8 du logiciel d'application.
- sauvegarde des registres du logiciel d'application dans la zone réservée à cet usage pour chaque tâche.
- analyse du code d'appel-système figurant dans la zone adresse de l'instruction CAL responsable de la demande.
- lancement de la fonction demandée.

Exécution de la fonction

Deux cas sont possibles :

- la fonction demandée est simple et ne nécessite pas de commutation. Elle est alors immédiatement traitée.
- la fonction demandée provoque une commutation.
  L'adresse de l'instruction CAL responsable de l'appel doit être sauvegardée afin de permettre le retour au bon endroit du logiciel d'application à la réactivation de la tâche.
  L'adresse d'appel est alors introduite sur la pile des appels associée à la tâche en début de traitement, puis dépilée au retour.
  La séquence de traitement de la fonction constitue donc alors un niveau sans en avoir l'appel standard.

Retour au service d'application

En plus des opérations propres au retour de fonction, cette séquence doit effectuer aussi le test de l'événement associé à l'opération de surveillance parallèle du terminal permettant de recevoir les demandes asynchrones de l'usager. Si l'événement est arrivé, le code correspondant est placé dans la Table du Contexte de la Tache (TCT), Si aucun code d'anomalie ne s'y trouve déjà; le traitement des anomalies est en effet plus prioritaire que les demandes de l'usager. Il pourra être obtenu ultérieurement par le service d'application, par exécution d'un appel-système spécial. L'opération de surveillance parallèle est alors relancée.

Les opérations suivantes sont donc exécutées au retour du traitement d'une fonction SAFRAN :

- test de la surveillance parallèle du terminal.
- restauration des registres à partir de la sauvegarde effectuée à l'appel de la fonction.
- retour derrière l'appel (l'adresse de retour a été calculée par la séquence de traitement de la fonction en fonction de la configuration de la table TIS propre à la fonction appelée).

Liste des niveaux simples d'exécution de fonctions

- Fonction forçage de commutation
  (remplacement du partage de temps en cas de séquences d'application faisant beaucoup de calculs.
  Elle se traduit par un simple appel du niveau : commutation avec gestion mémoire.)
- Fonction lecture au terminal
  Elle assure l'accès au terminal par appel au niveau : lecture au terminal avec gestion mémoire (la lecture s'effectue dans le tampon résident réservé à la tâche. Voir accès aux terminaux), et la recopie de l'information lue, dans la zone de travail de la séquence d'application.
  Si une anomalie est détectée pendant la lecture, un code est placé dans la table TCT. Ce code pourra être obtenu ultérieurement par le service d'application, par exécution d'un appel-système spécial.
- Fonction écriture au terminal
  Elle assure la recopie de l'information à transmettre dans le tampon résident, et l'accès au terminal par appel au niveau : écriture au terminal avec gestion mémoire.
  Les anomalies sont traitées comme pour une lecture au terminal.

Liste des fonctions simples au traitement immédiat

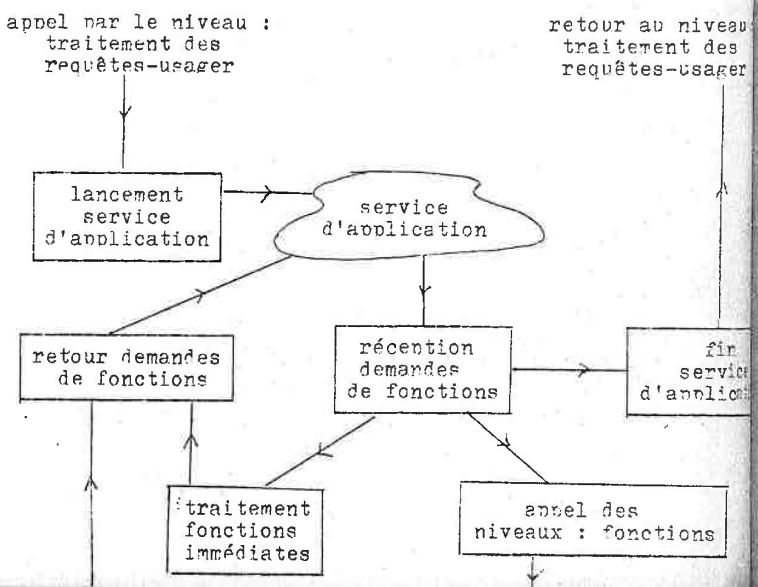
- Fonctions formatage des messages envoyés au terminal
  Ces fonctions se traduisent par des appels aux services TAM correspondants.
- Fonction obtention d'indicatif usager
  C'est un simple transfert d'information de la table TCT vers la séquence d'application.
- Fonction acquittement d'anomalie

C'est aussi un simple transfert d'information de la tab
TCT vers la séquence d'application.
Mais c'est cette fonction importante qui permet à la séq
ce d'application de prendre connaissance des anomalies d
cès aux terminaux, des demandes asynchrones lancées par
sager etc...

- Fonction enchaînement de services d'application
  Cette fonction se traduit par une simple mise en place d
  formations dans la table TCT. Lorsque ces informations s
  présentes, le niveau : traitement des requêtes-usager cou
  circuite le passage à l'état exécutif en fin d'exécution
  service d'application, détruit ces informations dans la
  ble TCT, et lance le service demandé en enchaînement.
- Fonction fin d'exécution
  Cette fonction se traduit par l'appel de la séquence ass
  rant l'achèvement de l'exécution du service d'applicatio
  et le retour au niveau : traitement des requêtes-usager.

Organisation générale du traitement des services d'applicat

L'organisation du niveau : exécution du logiciel d'applicat
peut se résumer dans le schéma suivant :



Fonctions de synchronisation

Principe [VIII] [IX]

Afin de permettre aux tâches gérées par le moniteur de se synchroniser entre elles, l'accès à des sémaphores est offert aux services d'application par des fonctions standard SAFRAN.

- La variable associée à un sémaphore peut être initialisée à une valeur quelconque, positive ou nulle, et ce au lancement de l'application ou à la suite d'un appel-système.
- Une file d'attente associée à chaque sémaphore permet de repérer les tâches bloquées derrière le sémaphore. Cette file d'attente étant gérée par un algorithme FIFO, les tâches bloquées sont réactivées dans l'ordre d'arrivée.

Deux fonction élémentaires d'accès à ces sémaphores sont offertes aux services d'application. Elles correspondent aux primitives P et V (Dijkstra).

Si u est la variable associée à un sémaphore :

- l'opération P répond à la logique suivante :
  Début
  u : = u-1 ;
  si u < 0 alors bloquer la tâche en cours en bout de file d'attente ;
  Fin ;
- l'opération V répond à la logique suivante :
  Début
  u : = u+1 ;
  si u ⩽ 0 alors débloquer la tâche bloquée en tête de file d'attente ;
  Fin ;

Réalisation d'un blocage de tâche

Dans certains cas, l'exécution d'une opération P provoque donc un blocage de la tâche. Ce blocage se traduit par une mise en attente de la tâche. La tâche doit alors pouvoir se réveiller.

- sous l'action d'une autre tâche (exécution de l'opération V).
- sous l'action d'une demande asynchrone de l'usager, mais une anomalie doit être signalée car le sémaphore n'est pas débloqué (ce reveil ne doit être utilisé que pour un désistement de la tâche face à la ressource protégée par le sémaphore).

Pour être libérable par demande asynchrone de l'usager, une tâche bloquée derrière un sémaphore doit être mise en attente de l'événement associé à la surveillance parallèle du terminal ceci peut être obtenu par demande de commutation avec réactivation conditionnelle. La libération de la tâche par une autre tâche, sans signal asynchrone de l'usager peut être obtenue par simple changement du type de réactivation : la tâche devenant immédiatement candidate à la réactivation n'est donc plus bloquée.

Implémentation des sémaphores

Chaque sémaphore doit contenir :

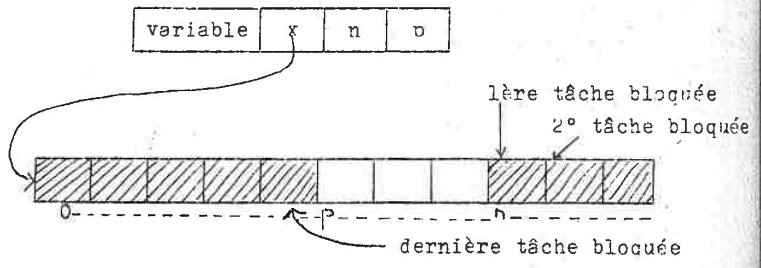
- la variable associée
- la file d'attente des tâches bloquées

La file d'attente peut contenir au maximum autant de tâches que le système peut en gérer moins une.
Une tâche bloquée est repérée dans la file par l'adresse de la table TCT associée.
La file d'attente étant gérée par l'algorithme FIFO, les ajouts se font en fin de file, les retraits se font en tête de file. Seuls les désistements, en principe rares, conduisent à un retrait en coeur de file.
Celle-ci peut donc être représentée par un tableau géré de façon circulaire.
La file peut alors se définir par les trois valeurs suivantes :

- adresse du tableau contenant la file d'attente
- index de la 1ère tâche de la file dans le tableau
- index de la 1ère position libre dans le tableau après la file.

Remarque : le tableau étant géré de façon circulaire, l'index de la première position libre peut être inférieur à celui de la 1ère tâche bloquée.

Le descripteur de sémaphore peut se schématiser comme suit :



L'introduction d'une tâche dans la file se traduit par les opérations suivantes :

- mise en place de l'adresse de la table TCT correspondante dans l'emplacement désigné par l'index de la 1ère position libre.
- mise à jour de l'index de la 1ère position libre (incrémentation de 1 ou retour à 0).

Le retrait d'une tâche de la file se traduit par l'opération suivante :

- mise à jour de l'index de la 1ère tâche bloquée (incrémentation de 1 ou retour à 0).

Traitement des désistements

Une tâche qui se désiste face à un sémaphore annule l'opération P qu'elle avait effectué auparavant.
Ceci nécessite :

- que la tâche connaisse le sémaphore qui la bloque (un pointeur doit donc exister dans la Table du Contexte de la Tâche (TCT) vers le descripteur du sémaphore bloquant. Ce pointeur est mis à jour à chaque blocage.
- que la variable associée au sémaphore bloquant soit restaurée (elle avait été décrémentée de 1).
- que l'adresse de la table TCT associée à la tâche soit extraite du coeur de la file d'attente du sémaphore. La file d'attente doit alors être tassée.
- un désistement étant l'annulation d'une opération P, et non une opération V, aucune tâche ne doit être débloquée.

Un désistement se traduisant par une relance de la tâche, le service d'application doit s'assurer, après une opération P, qu'aucun désistement n'a eu lieu (acquittement de l'anomalie).

Traitement des primitives de synchronisation

Les tâches n'étant interruptibles qu'à des points précis choisis par elles, l'exclusivité d'accès à la variable associée au sémaphore est réalisée de fait, sans précautions supplémentaires. Il n'en serait pas de même en cas de Time Slicing.
Si u est la variable associée au sémaphore, l'algorithme peut être formalisé comme suit :

Traitement de la primitive P :

```
Début
u : = u-1 ;
si u < 0 alors Début
            Ajouter index tâche en cours en fin de file ;
            fin de file : = tâche en cours ;
            Commutation avec réactivation conditionnelle ;
            si Réactivation normale
                  Alors Début
                        Tête de file : = suivant (tête de
                        file) ;
                        Fin
                  Sinon Début
                        Sortir index tâche en cours du co
                        de la file ;
                        u : = u+1 ;
                        Fin ;
            Fin ;
Fin ;
```

Remarque : si les deux types de déblocage de la tâche se produisent simultanément, c'est bien sûr un déblocage normal qui est détecté, avec anomalie dûe à la transmission d'un caractère Attention (et non avec anomalie déblocage par l'usager).

Traitement de la primitive V :

```
Début
u : = u+1 ;
si u ⩽ 0 Alors Début
            tâche à réveiller : = tête de file ;
            déclarer tâche à réveiller immédiatement candi-
            date à l'activation ;
            Fin ;
Fin ;
```

ADAPTATION DU MONITEUR SAFRAN A UNE APPLICATION

Conception du logiciel d'application

Généralités

Le moniteur SAFRAN n'est qu'un noyau de système sans application propre. Il ne peut être utilisé qu'avec l'adjonction de logiciels d'application. La conception d'une application SAFRAN se réduit généralement à la mise au point de ces services.
Le logiciel d'application est formé de un ou plusieurs programmes en langage machine IRIS 80, décrivant chacun un algorithme de bout en bout, comme s'ils travaillaient pour le compte d'un seul usager. Le partage de ces programmes, géré par le moniteur, est pratiquement transparent : seules quelques règles simples de programmation sont imposées. Ces programmes ont accès aux fonctions offertes par le moniteur SAFRAN, à condition de respecter l'interface d'appel. Certaines fonctions offertes par SIRIS 8 leur sont aussi accessibles.
Au lancement de l'application, ces programmes sont contenus dans un fichier sur disque appelé : fichier des services. Ils sont chargés en mémoire par le moniteur SAFRAN en fonction des demandes d'exécution lancées par les usagers.

Zone mémoire réservée au logiciel d'application

Afin de permettre le chargement en mémoire des séquences d'application pour exécution, toute application SAFRAN doit disposer d'un espace-mémoire libre. Cet espace est logiquement divisé en deux zones distinctes : un zone est destinée à recevoir la partie code des programmes (texte), l'autre est réservée au chargement des variables. Ces deux zones sont gérées de façon

différente par le moniteur SAFRAN (voir commutation). Un seul service étant chargé à la fois dans cet espace, les textes des programmes sont toujours chargés à la même adresse, de même que les zones de variables.

Il en résulte que :

- la zone texte doit être assez grande pour contenir la plus longue des séquences d'application.
- la zone des variables doit être assez grande pour contenir la plus longue des zones de variables utilisées par les séquences d'application.
- tous les services d'application doivent respecter la même implantation.

En pratique, dans toute application SAFRAN, la zone réservée aux variables d'application commence en début de la mémoire virtuelle de l'application (BIAS), la zone réservée aux textes des services est située immédiatement derrière.

Règles imposées aux séquences d'application

1- Les variables doivent être groupées dans un espace mémoire qui ne sera jamais initialisé au chargement.
2- Le code et les données fixes du programme doivent être groupés dans un espace mémoire qui ne devra jamais être modifié.
3- Les adresses référencées par le programme doivent être ajustées afin de permettre une exécution correcte lorsque le programme sera chargé dans les zones correspondantes réservées dans l'application.
4- La première instruction exécutée pour un service est toujours contenue dans le premier mot de la zone texte (adresse de départ).
5- La structure des tables TIS fournies par le programme doit être en accord avec l'interface attendue par les séquences du moniteur traitant les appels-système.
6- Les séquences de calcul contenues dans les services d'application doivent être régulièrement interrompues par des appels-système provoquant la commutation.
7- Les séquences d'application ont accès à la majorité des appels-moniteur standard SIRIS 8 sous réserve que tous les paramètres d'appel soient contenus dans la zone texte du programme.

Pour qu'un programme respectant les règles précédentes soit exécutable sous le contrôle du moniteur SAFRAN, le texte de ce

programme doit être introduit dans un fichier disque standard SIRIS 8 d'accès direct.

Les trois informations suivantes devront ensuite être introduites dans la table TDP :

- l'adresse dans le fichier du premier bloc contenant le texte du programme.
- la taille en octets de la zone texte.
- la taille en octets de la zone des variables.

Remarque : respect de la règle 6 :

La longueur maximale des séquences de calcul admissibles entre deux appels-système provoquant la commutation, doit être évaluée en fonction de la charge prévue de l'application, et du temps de réponse souhaité.

Programmation pratique des séquences d'application

Tout le logiciel nécessaire pour réaliser facilement la création d'un fichier de services à partir de programmes écrits en langage symbolique METASYMBOL est actuellement disponible :

- les règles 1, 2, 4, 6 sont de simples disciplines de programmation.
- des procédures de méta-assemblage permettent de décrire simplement l'implantation souhaitée pour le programme (règle 3).
- des procédures de méta-assemblage permettent de générer facilement les instructions d'appels-système et les tables TIS associées (règle 5).
- la plupart des services SIRIS 8 sont directement accessibles par les programmes d'application. Certains, en particulier ceux qui utilisent des informations en zones mémoires protégées, ne sont pas directement accessibles par les séquences d'application (règle 7).
- la recopie du texte du programme sur le fichier est assurée par une procédure disponible qui sera simplement ajoutée au programme.

La marche à suivre pour constituer le fichier est décrite en annexe B.

Génération du système

Paramètres variables

Le système SAFRAN a été conçu pour être au maximum indépendant des applications qu'il est amené à contrôler. Mais toutes les applications ne présentent généralement pas les mêmes caractéristiques : un certain nombre de paramètres sont donc différents d'une application à l'autre.
Les principales variables sont les suivantes :

- le réseau de terminaux :
  le nombre des terminaux
  la taille maximale des messages envoyés aux terminaux (longueur de ligne).
- le logiciel d'application :
  les noms et tailles des processeurs (services d'applications).
- la partition mémoire réservée au logiciel d'application :
  la taille de la zone des variables
  la taille de la zone texte
- les sémaphores utilisés :
  le nombre des sémaphores
  les valeurs initiales des variables associées
- les fichiers utilisés :
  la taille des blocs du fichier swapping
  la taille des blocs du fichier des services
- le nom de l'application (plusieurs applications SAFRAN peuvent s'exécuter simultanément sur le même site).

Incidence des paramètres sur le moniteur

- Le nombre de terminaux
  Il fixe le nombre maximal de tâches gérées par le système. Toutes les informations propres aux tâches sont donc en nombres variables. Parmi les plus importantes, il faut citer :
  les tables TCT
  les piles des appels
  les tampons résidents
  les zones de sauvegarde de registres
- le logiciel d'application
  Il fixe :
  la taille de la partition mémoire réservée au logiciel d'application.
  la longueur et le contenu de la table TDP.

- Le nombre de sémaphores utilisés
  Il fixe le nombre de descripteurs de sémaphores et de files d'attente .
- Les tailles des blocs des fichiers utilisés
  Ce sont des paramètres des tables DCB SIRIS 8 utilisées par l'application.
- Le nom d'application et la taille maximale des messages envoyés au terminal sont contenus dans la table TRG.

En résumé, d'une application à l'autre, seules les tables changent. Les algorithmes, donc le texte du système, sont les mêmes. Pour adapter le moniteur à une application, il suffit donc de créer une configuration de tables conforme aux paramètres propres à l'application.

Adaptation à une configuration

Deux solutions sont possibles :

- créer une version adaptée à chaque application par génération d'un programme source et assembler. Cette solution crée une version d'un lancement rapide, mais nécessitant une nouvelle génération à chaque modification (addition d'un terminal, par exemple). Les modifications éventuelles de structure des tables sont faciles.
- utiliser un système auto-adaptable créant dynamiquement ses tables au lancement, en fonction d'un fichier de paramètres (des services SIRIS 8 permettent l'allocation dynamique de mémoire). Cette solution est souple, car elle ne nécessite pas de réassemblage pour une modification, seulement un changement du fichier des paramètres. Par contre, elle nécessite une lourde séquence au lancement, et rend plus délicate la modification de la structure des tables.

La version actuelle du moniteur SAFRAN utilise la première méthode.

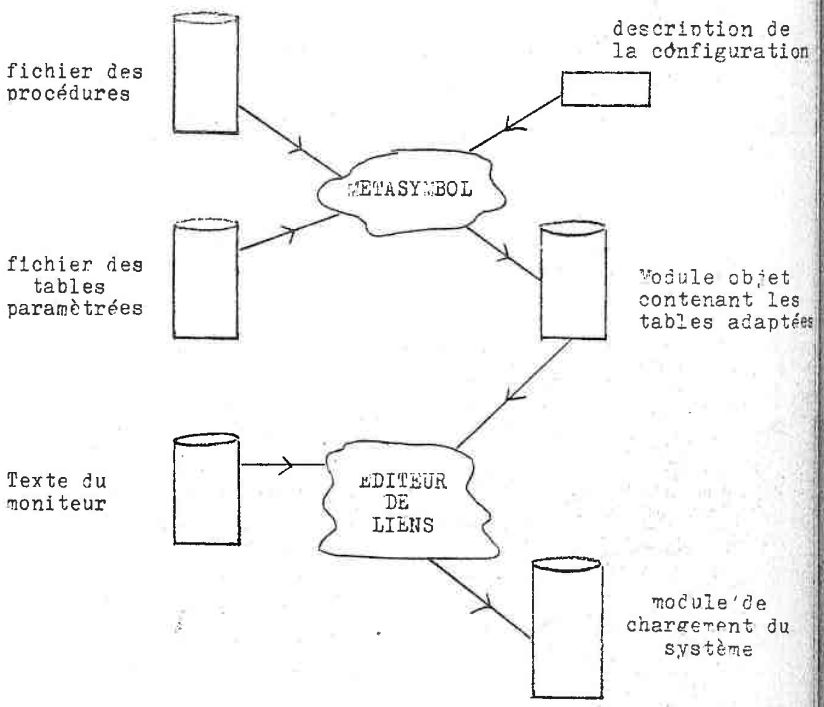
Déroulement de la génération [XI]

- premier temps : construction des tables
  Cette opération utilise les services du méta-assembleur METASYMBOL.
  Un fichier symbolique contient la description des tables paramétrée par l'usage de méta-variables.
  Un autre fichier symbolique contient des procédures permet-

tant d'affecter facilement des valeurs à ces méta-variables.
La description d'une configuration se réduit donc à une suite d'appels de procédures METASYMBOL.
L'assemblage du programme constitué par
l'appel du fichier symbolique des procédures
la description de la configuration
l'appel du fichier symbolique contenant la description paramètrée des tables
permet de constituer un module-objet contenant les tables adaptées à la configuration.

- deuxième temps : construction du module de chargement
L'édition des tables créées à l'étape précédente et du moniteur permet de créer le module de chargement du système.
Les tables doivent être éditées avant le moniteur afin d'assurer à la zone réservée au logiciel d'application une localisation cadrée à la BIAS du système.

Le déroulement exact des opérations est décrit en annexe D.
Le schéma suivant résume l'opération de génération :



Extensions

Limites du système standard

Les opérations traitées par le moniteur SAFRAN de base sont très limitées. Certaines applications ne peuvent pas être facilement mises au point à cause de ces limites.

- un exemple :
Lorsque des services d'application sont en cours pour le compte de plusieurs usagers, ils n'ont pas de moyen simple pour échanger des informations entre eux. La seule solution accessible est de passer par des boîtes aux lettres SIRIS 8.
- un autre exemple :
Certains appels-moniteur SIRIS 8 nécessitent de nombreux paramètres. Lors de l'écriture d'un programme en langage symbolique, des macro-instructions permettent d'engendrer facilement ces appels ; les informations correspondantes sont groupées dans des tables (FPT) construites par METASYMBOL dans des sections de contrôle protégées, donc ne faisant pas partie de celle du texte du programme.
Pour que les services d'application puissent accéder à ces fonctions, les tables FPT et les appels-moniteur doivent être programmés dans des sections non protégées.

Points d'extensions

Il est donc souhaitable de permettre au concepteur d'une application SAFRAN d'insérer des séquences à tous les points-clés du système :

- à la demande de connexion d'un usager (début de session).
Cette séquence peut être utilisée par exemple pour établir un dialogue avec l'usager avant le début du travail, afin de vérifier son identité (mot de passe), d'initialiser une comptabilité, éventuellement de refuser la connexion etc... (opération de LOGIN).
- à la demande de déconnexion d'un usager (fin de session).
Cette séquence peut être utilisée par exemple pour effectuer des sauvegardes d'information, des copies de fichiers etc...
- à l'exécution d'un service d'application (ajouts de fonctions appelées par de nouveaux appels-système).
Ces séquences peuvent être utilisées pour effectuer des opérations internes au moniteur SAFRAN non prévues dans

le logiciel standard, pour offrir aux services d'application l'accès à des fonctions de SIRIS 8 qu'ils ne pourraient appeler autrement (tous les services de SIRIS 8 sont accessibles au moniteur SAFRAN) etc...

- au lancement et à la fin d'un service d'application
  En fait, le concepteur de l'application étant maître de tous les services offerts aux usagers, ces opérations peuvent être traitées par des appels-système exécutés dès le lancement, ou juste avant le retour du processeur.
- à l'exécution des services standard (ajout de nouvelles commandes).
  Ces séquences peuvent être utilisées lorsque le concepteur désire qu'un service soit toujours résident, ou s'il s'agit d'un service très simple non traitable comme service d'application (même cas que les appels-système).
- à l'initialisation du système
  Pour initialiser les autres services ajoutés...

Bien sûr, les séquences standard du moniteur ne doivent pas être modifiées par ces insertions.
De telles séquences, considérées cependant comme des séquences du moniteur, doivent respecter les mêmes règles que le reste du moniteur, et respecter l'interface définie pour elles. L'annexe C décrit le moniteur, et son interface avec les séquences ajoutées.
En pratique, de telles séquences sont programmées de façon indépendante, et insérées à la génération de système.
L'annexe D décrit la marche à suivre pour effectuer pratiquement ces insertions.

Insertion de séquences au lancement

Pour qu'une séquence ajoutée soit exécutée au lancement du système, il faut :

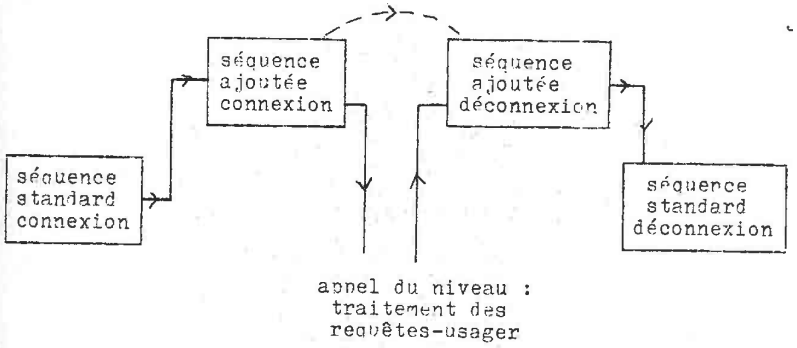
- que la séquence standard de lancement contienne une instruction de branchement à la séquence ajoutée.
- que la séquence ajoutée contienne une instruction de branchement vers une adresse de retour dans la séquence standard

Interface à l'appel :
Pour que la même séquence standard soit valable dans tous les cas, une table construite à la génération du système doit contenir l'adresse de la séquence ajoutée.

Une procédure ETASYMBOL permet de déclarer cette adresse à l'étape de construction des tables.
Interface au retour :
c'est une simple règle imposée à la séquence ajoutée.

Insertion de séquences à la connexion et à la déconnexion

Ces séquences font logiquement partie du niveau : traitement des connexions et déconnexions. Elles sont exécutées respectivement juste avant et juste après l'appel au niveau : traitement des requêtes usager.
L'insertion suit le même principe que celle de la séquence ajoutée au lancement.
N.B. : si la séquence ajoutée à la connexion effectue un appel à la séquence ajoutée à la déconnexion, la connexion sera de fait refusée.



Ajouts d'appels-moniteur et de commandes immédiates

La séquence destinée à reconnaître les appels-moniteur (voir traitement des requêtes-processeur) utilise une table faisant correspondre au code d'appel-moniteur, l'adresse de la séquence correspondante.
La séquence destinée à reconnaître les services standard ou commandes immédiates (voir traitement des requêtes-usager) utilise une table faisant correspondre au nom de la commande, l'adresse de la séquence correspondante.
Pour ajouter de nouvelles fonctions offertes au logiciel d'exploitation et de nouveaux services offerts à l'usager, il suffit d'ajouter des entrées à ces tables.

A la génération du système, des procédures METASYMBOL permettent de déclarer les adresses correspondantes, à l'étape de construction des tables.

Information ajoutée

- les extensions apportées au moniteur SAFRAN peuvent faire appel aux niveaux standard (commutations, accès au terminal etc...). Elles peuvent aussi se diviser elles-même en plusieurs niveaux. La pile standard des appels peut ainsi être insuffisante.
  Une procédure METASYMBOL permet de demander à la génération un allongement de la pile des appels (à l'étape de construction des tables).
- les extensions apportées au moniteur SAFRAN peuvent nécessiter des variables de travail. Ces variables peuvent être :
  communes : l'espace mémoire peut alors leur être réservé avec les textes ajoutés.
  propres : (en cas de conservation de données pendant les commutations).
  L'espace alloué à chaque tâche doit alors lui être directement accessible.
  Pour cela, l'adresse de l'espace correspondant doit être repéré par un pointeur dans la Table des Contextes de Tâches (TCT).
  Une procédure METASYMBOL permet de demander à la génération du système, la réservation d'une zone propre à chaque tâche.
  (Les séquences ajoutées peuvent accéder aux tables TCT grâce à l'index de reentrance).

Problème de l'accès aux fichiers

Enoncé du problème [XII]

Lorsque les services d'application doivent accéder à des fichiers en création, lecture, destruction etc, ces services doivent pouvoir effectuer les opérations suivantes :

- création de DCB's
- assignation de fichiers réels à ces DCB's
- ouvertures
- lectures, écritures
- fermetures

Or les contraintes suivantes sont imposées par le système de gestion de fichiers :

- les DCB's sont généralement créés en zones protégées. Il est très imprudent de ne pas respecter cette règle, et qui plus est, de les ranger dans des zones non-résidentes.
- l'appel-moniteur effectuant l'assignation entre dans la catégorie décrite au chapitre précédent (génération de plusieurs sections).

L'appel-moniteur d'assignation peut, à la limite, être programmé directement. Les DCB's peuvent être créés dynamiquement en zone protégée et résidente par des appels-moniteur SIRIS 8 spécialisés, à chaque demande de service, mais cette méthode conduit peu à peu à la saturation de la mémoire, et à l'avortement de l'application pour manque d'espace.

Solution proposée

Les extensions au moniteur permettent de résoudre ce problème :

- les DCB's nécessaires peuvent être créés au lancement de l'application par une séquence ajoutée à l'initialisation. En effet, les DCB's étant réutilisables, il suffit d'en créer, pour chaque tâche, autant qu'il pourra y en avoir d'ouverts pour la tâche à un instant donné.
  Les adresses de ces DCB's peuvent être conservées dans une zone propre ajoutée à chaque tâche.
- les services d'application peuvent obtenir les adresses de ces DCB's et effectuer les assignations par l'appel d'une fonction spécialement ajoutée.
- disposant des adresses des DCB's, ils peuvent effectuer les opérations d'ouverture, d'accès, de fermeture, par appels des services standard SIRIS 8.

BILAN ET PERSPECTIVES

Etat actuel du moniteur SAFRAN

Points positifs

Une première version du moniteur SAFRAN est actuellement opérationnelle. A titre expérimental, elle a été mise en oeuvre dans les conditions suivantes :

- l'application d'essai est du type : éditeur de textes, donc très conversationnelle et effectuant peu de calculs.
- un petit nombre de terminaux ont accès à ses services (6).

Si cette application est lancée avec une priorité d'exécution suffisante, on constate :

- que la réponse est quasi-immédiate malgré l'encombrement mémoire très réduit (12 pages).
- que la charge du système SIRIS 8 semble fort peu retentir sur ce temps de réponse, ce qui ne serait pas le cas si les terminaux étaient en liaison avec le système Temps Partagé de SIRIS 8.

Les simulateurs de machines décrits précédemment présentent des caractéristiques d'exploitation voisines de l'application d'essai. Il semble donc que le moniteur SAFRAN soit bien adapté à leur gestion dans le cadre de sessions de travaux pratiques en groupe restreints. Il garantit en effet à peu de frais un bon temps de réponse : la consommation de ressources (temps de calcul, mémoire, transferts-disques) est alors très limitée.

Points incertains

- Etant donné le faible nombre de terminaux disponibles pour les tests sur le site expérimental, il est difficile de prédire le comportement du moniteur SAFRAN s'il doit gérer un important réseau de terminaux.

Le nombre maximum de terminaux raisonnablement connectables à une application est d'ailleurs probablement très variable, en fonction des caractéristiques de l'application. Interviennent en effet :

- la vitesse des terminaux :
  pendant l'édition d'une même ligne de caractères, plus le terminal est lent, plus le moniteur peut gérer de tâches sans freinage apparent.
- la puissance de calcul utilisée :
  plus les séquences de calcul sont courtes entre les Entrées-Sorties, plus le moniteur peut gérer de tâches sans freinage apparent, pendant la durée d'un accès au terminal.

Il semble donc intéressant d'entreprendre l'étude d'un modèle du comportement du moniteur SAFRAN, afin d'étudier les réactions de ce moniteur dans divers cas d'applications possibles. Ceci permettrait de connaître ses limites, et éventuellement de proposer des améliorations.

- S'il est facile de constater qu'une petite application SAFRAN, sans interférence avec le système Temps Partagé actif sur le même site, a meilleure réponse que la même application ajoutée au système Temps Partagé déjà chargé, le résultat d'une confrontation entre ces deux moniteurs à charge égale et dans des conditions identiques (mémoire équivalente) est très incertain. En effet :
  - le moniteur SAFRAN n'exerce aucun contrôle sur les tâches qu'il gère, mais, étant lui-même travail-usager, il subit les contrôles de SIRIS 8.
  - le moniteur Temps Partagé fait des contrôles sur les tâches qu'il gère, mais, étant lui-même tâche système, il ne subit pas de contrôles de SIRIS 8.
  - le moniteur SAFRAN optimise le nombre des swappings (partage des textes des processeurs) mais le moniteur Temps Partagé a accès à des Entrées-Sorties prioritaires sur disque.

Une étude comparative efficace "au chronomètre" sur une application test donnerait des indications, mais nécessiterait une charge importante des moniteurs, donc un nombre élevé de terminaux.
Une étude théorique imposerait un travail très important d'analyse du moniteur SIRIS 8, et pour cela, l'accès à des documents confidentiels.

Le doute risque donc de peser longtemps sur ce point.

Points négatifs

L'accès d'un travail à des terminaux par la méthode TAM impose la réservation de ces terminaux à ce travail. (C'est le cas pour toutes les assignations de périphériques du type DEVICE).
L'utilisation d'un terminal par une application SAFRAN impose donc son retrait de la liste des terminaux gérés par le moniteur Temps Partagé. Dès lors, l'usager du terminal ne peut plus avoir accès qu'aux services offerts par cette application, et ce jusqu'à son arrêt définitif.
De plus, toutes ces opérations nécessitent l'intervention de l'opérateur central.
Ces restrictions, inhérentes à la méthode d'accès TAM, ne peuvent être contournées. Heureusement, la versions C10 du système SIRIS 8 apporte une nouvelle méthode d'accès aux terminaux (VCAM) permettant à ces terminaux d'obtenir à leur initiative une mise en contact avec une application de leur choix (Temps Partagé ou n'importe quelle autre application utilisant les services de la méthode d'accès VCAM), et ce sans intervention de l'opérateur central.
Cette restriction est donc tout à fait temporaire.

Evolution prévisible à court terme

Principe

La version C10 du système SIRIS 8 apporte de nouveaux services d'accès aux terminaux : avec le développement de nouvelles configurations de réseaux, l'accès d'une application à des terminaux en passant par frontaux, noeuds, concentrateurs, necessite la prise en compte des protocoles propres au réseau. La méthode d'accès VCAM prend en charge ces protocoles, et donne aux applications un moyen unique d'accéder aux terminaux du réseau, quelle que soit leur localisation.
De plus, l'ouverture au réseau, des services d'une application travaillant avec cette méthode d'accès ne nécessite plus la réservation exclusive des terminaux, leur laissant ainsi leur totale liberté de choix des connexions.
Il est donc souhaitable de remplacer, dans le moniteur SAFRAN, les appels à la méthode d'accès TAM par des appels à la méthode

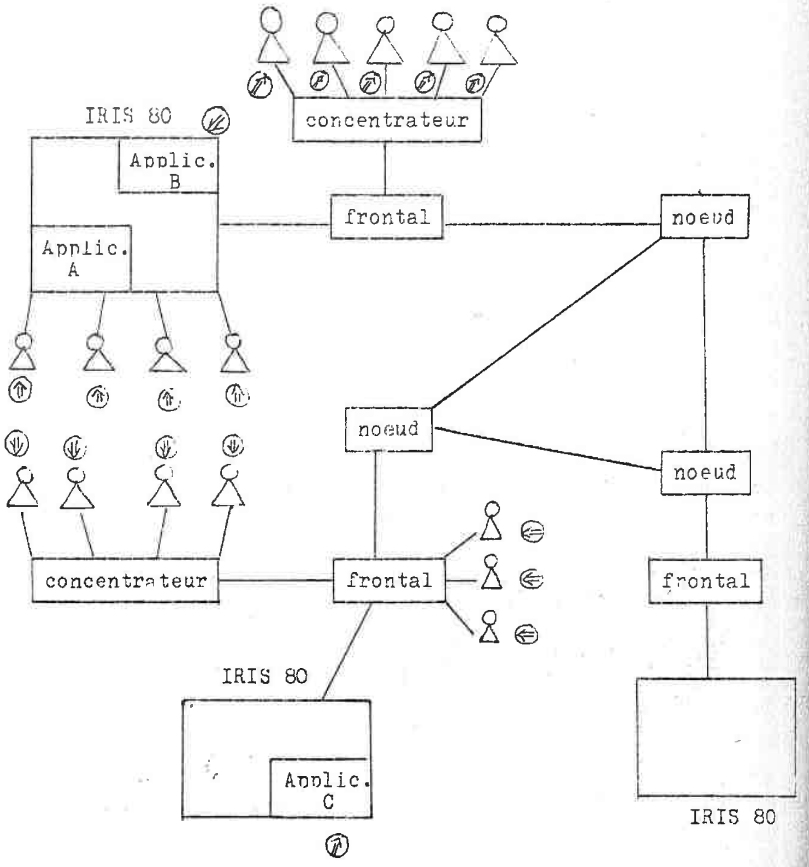
d'accès VCAM, et ce dans les plus brefs délais.

Ouverture des applications SAFRAN aux nouveaux réseaux [XIII]

La méthode d'accès VCAM (Virtual Communications Access Method) a été conçue pour permettre des échanges d'informations entre :

- plusieurs applications (sur le même ordinateur ou sur des ordinateurs distincts).
- des applications et des terminaux (directement connectés à l'ordinateur ou reliés à cet ordinateur par le réseau).

Afin de permettre une mise en oeuvre facile des applications, une interface commune a été définie, la même pour toutes les communications. Une application peut ainsi entrer en contact avec des correspondants sans connaître ni la nature, ni la localisation de ces correspondants.



Sur ce schéma, les signes (⇒) mettent en évidence des correspondants possibles d'une application A active sur un ordinateur d'un réseau.

Ces correspondants peuvent donc être :

- d'autres applications sur le même ordinateur.
- des terminaux directement connectés sur cet ordinateur.
- des correspondants distants dont les communications passent par un ordinateur frontal.

Notion de correspondant virtuel

En pratique, tout correspondant du réseau, terminal ou application, désirant ou acceptant d'entrer en communication avec d'autres correspondants du réseau, terminaux ou applications, doit se déclarer au réseau pour être accessible. Un identificateur unique dans le réseau lui est alors affecté.

On appelle site un élément du réseau où est tenu à jour un catalogue des identifications des correspondants auxquels il assure l'accès au réseau.

Un ordinateur IRIS 80 est un site (central). Les correspondants dont il assure la gestion peuvent être des applications ou des terminaux directement connectés.

Un frontal est un site. Les correspondants dont il assure la gestion ne peuvent être que des terminaux qui lui sont connectés (directement ou par un concentrateur).

Le rôle du réseau est de mettre ces correspondants en communication et de transferer entre eux des informations, que ces correspondants soient sur le même site ou non : des logiciels doivent être disponibles sur tous les sites pour cet usage.

Pour chaque correspondant d'un réseau, l'ensemble des correspondants virtuels possibles est la liste des identificateurs de tous les correspondants déclarés au reseau.

Gestion des correspondants du réseau au niveau d'un site central

Tout ordinateur IRIS 80 connecté à un réseau de ce type est un site. Le système SIRIS 8 doit donc tenir à jour un catalogue des identifications des correspondants dont il assure la gestion.

Ces correspondants peuvent être :

- des terminaux directement connectés
- des applications en exécution sur cet ordinateur.

En exploitation courante, les terminaux conversationnels directement connectés sont gérés par une tâche système (TGT). Dès son lancement par l'opérateur central, cette tâche déclare la liste des terminaux choisis comme correspondants du réseau. Leurs identifications sont introduites dans le catalogue.
Des applications peuvent se déclarer correspondants du réseau et demander (ou accepter) des communications avec d'autres correspondants (sur n'importe quel site) grâce aux services de la méthode d'accès VCAM.

Mise en oeuvre de VCAM pour le dialogue avec un terminal

Les principales fonctions offertes par VCAM sont les suivantes :

- déclaration de l'application comme correspondant du réseau.
- demande de mise en communication avec un correspondant.
- acceptation d'une demande de mise en communication lancée par un correspondant.
- refus d'une telle demande.
- émission d'une information vers un correspondant en communication.
- réception d'une information émise par un correspondant en communication.
- cessation de l'activité correspondant réseau.

Ces fonctions sont traitées par des services de deux types :

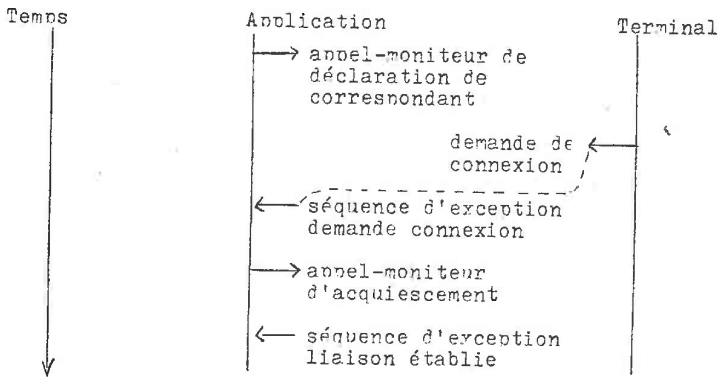
- les services offerts par appels-moniteur SIRIS 8
  Ils permettent au programme de demander à VCAM l'exécution d'opérations pour son compte.
- les services offerts par l'activation de séquences d'exception.
  Ils permettent au programme d'être averti d'évènements asynchrones. Plusieurs types de séquences d'exception, correspondant à différents types d'évènements, peuvent ainsi être activées.

Cette interface est commune à tous les correspondants du réseau. En pratique, certains correspondants peuvent n'autoriser qu'un sous-ensemble des possibilités d'échanges offertes par le réseau. Les limites des échanges possibles doivent être convenues entre les correspondants dès l'établissement de la communication.

Mise en communication d'une application et d'un terminal

Le terminal doit avoir été déclaré correspondant du réseau.
Tous les organes du réseau nécessaires pour y accéder doivent être opérationnels.
L'opération doit aussi s'être déclarée correspondant du réseau.

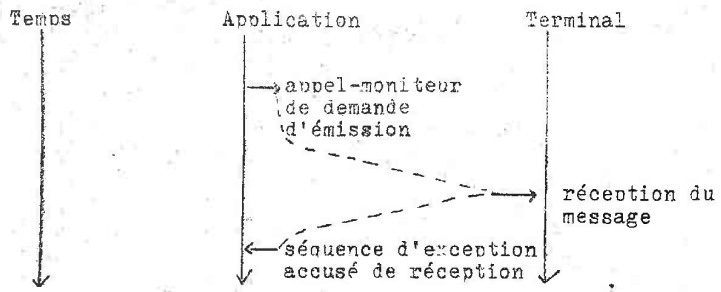
- La mise en communication se fait à l'initiative du terminal ; une demande de connexion est transmise à l'application.
- L'application est déroutée en séquence d'exception, afin d'être informée de la demande du terminal.
- L'application peut, si elle accepte la communication, exécuter un appel-moniteur d'acquiescement. Sinon, elle doit exécuter un appel-moniteur de refus
- Après l'établissement de la communication par le réseau, l'application est de nouveau déroutée pour être informée de la connexion.



Dialogue de l'application avec le terminal

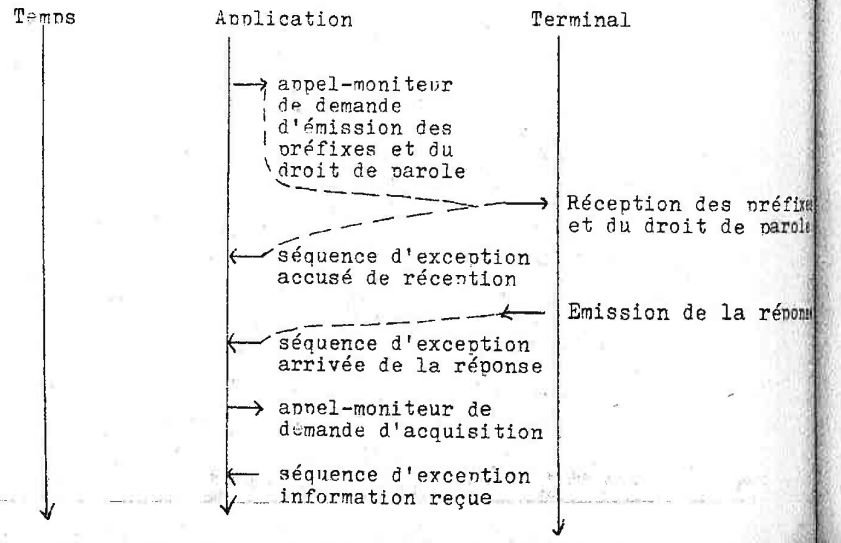
Le dialogue de l'application avec le terminal se fait en mode alternat.
Au départ, l'application a le droit de parole : elle peut émettre librement des informations vers le terminal.
L'émission se fait en deux temps :

- demande d'émission lancée par l'application (appel-moniteur).
- déroutement en séquence d'exception après transmission pour retour d'accusé de réception indiquant les conditions de transfert (erreurs etc...)



Pour effectuer une saisie d'information au terminal, l'application confie le droit de parole à ce terminal. Celui-ci restitue le droit de parole avec le message introduit.
La lecture se fait en cinq temps :

- demande d'émission lancée par l'application (appel-moniteur). Cette opération transfère au terminal les caractères préfixes de lecture et le droit de parole.
- déroutement en séquence d'exception (retour d'accusé de réception du droit de parole).
- déroutement en séquence d'exception à l'arrivée de l'information émise par le terminal.
- demande d'acquisition de l'information lancée par l'application et du droit de parole (appel-moniteur).
- déroutement en séquence d'exception indiquant la mise à la disposition de l'information.



Transmission des caractères Attention émis par le terminal

L'émission d'un caractère Attention au terminal provoque l'activation dans l'application d'une séquence d'exception. Cette opération n'est pas soumise au droit de parole, et ne perturbe pas les échanges ordinaires entre l'application et le terminal.

Utilisation de la méthode d'accès VCAM dans le moniteur SAFRAN

Pour permettre une transformation facile du moniteur SAFRAN, il faut construire une interface utilisant les services VCAM et offrant des services identiques à ceux offerts par l'interface TAM. Il suffit alors d'un échange standard.
Cette interface est formée de séquences de deux types :

- des sous-programmes présentant la même interface que les services TAM, qui seront appelés par le moniteur SAFRAN dans les mêmes conditions.
- des séquences d'exception dont l'exécution est transparente au moniteur SAFRAN (activées sur interruption, et restituant au retour le contexte du travail avant l'appel), et qui permettent les postages asynchrones des événements.

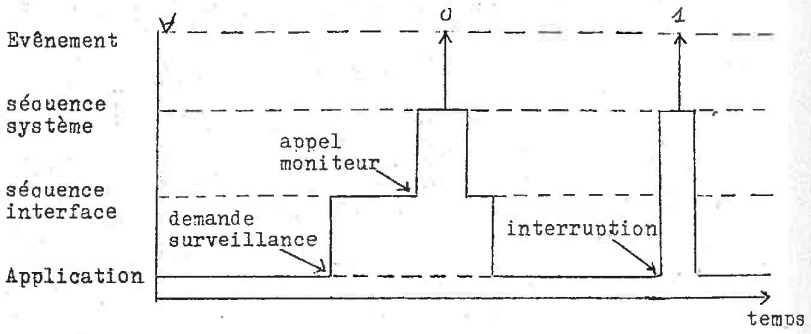
N.B. : ceci n'est possible que parce que les appels-moniteur VCAM peuvent être exécutés pendant des séquences d'exception.

Surveillance du terminal avant la connexion

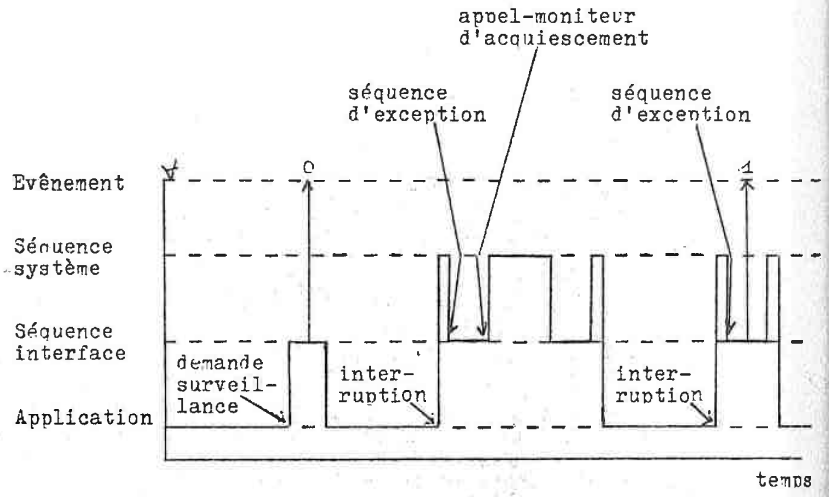
- La procédure équivalente a pour seul rôle d'initialiser un événement.
- La séquence d'exception associée à la demande de connexion du terminal est transparente à l'application. L'appel-moniteur d'acquiescement est exécuté dans cette séquence.
- La séquence d'exception activée à l'établissement de la liaison est utilisée pour poster l'événement.

Les schémas suivants décrivent le déroulement des séquences application, interface d'accès, et système dans les deux cas TAM et VCAM.

TAM :



VCAM :



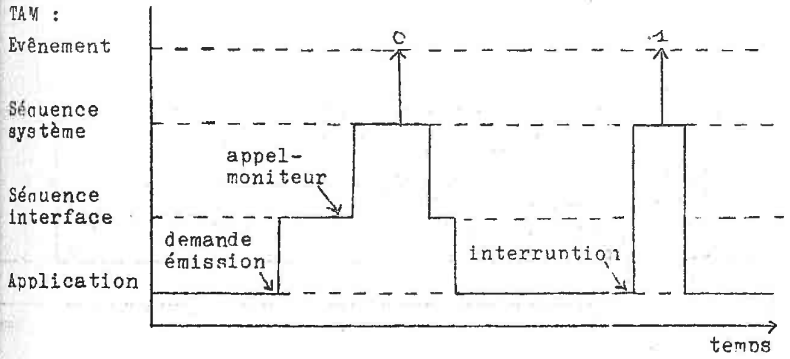
La surveillance parallèle est traitée de même, mais la séquence d'exception utilisée pour poster l'évênement est la séquence activée sur émission d'un Attention.

Emission de messages

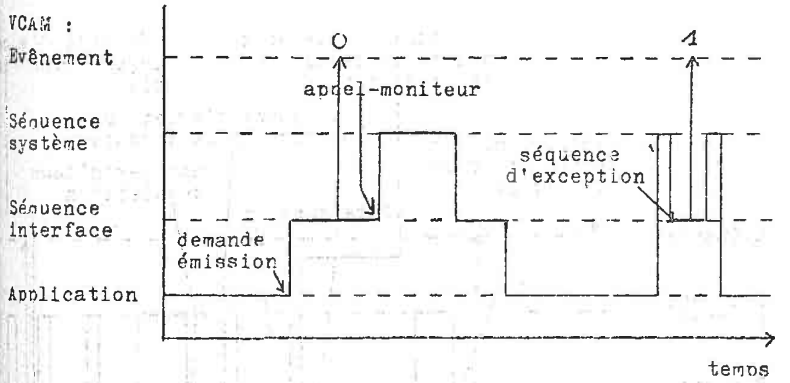
- La procédure équivalente a pour rôle d'effectuer la demande d'émission et d'initialiser l'évênement.
- La séquence d'exception associée à l'accusé de réception est

utilisée pour poster l'évênement.

TAM :



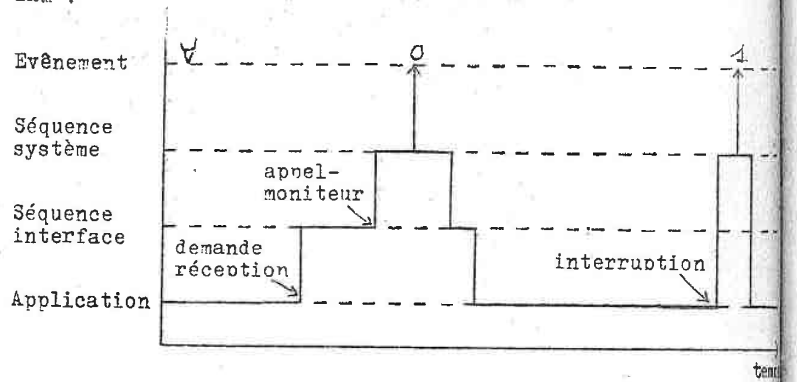
VCAM :



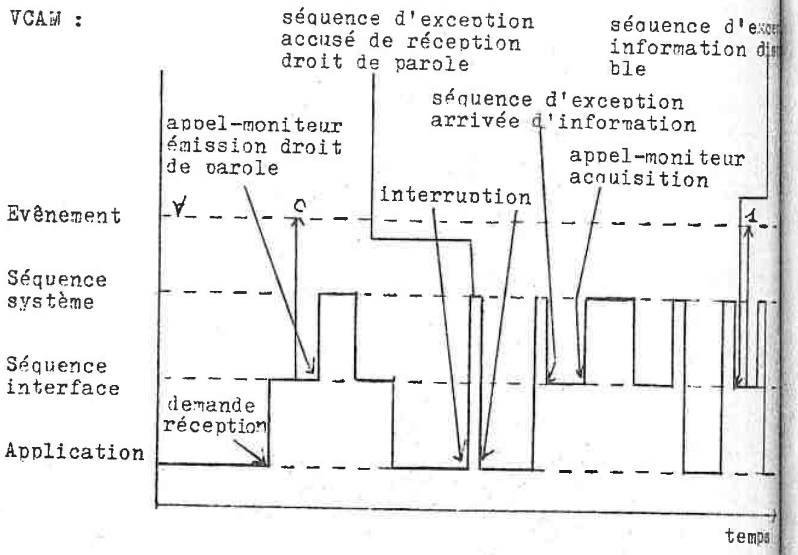
Réception de messages

- La procédure équivalente a pour rôle d'effectuer la demande de transfert du droit de parole et d'initialiser l'évênement.
- La séquence d'exception associée à l'accusé de réception est inutilisée.
- La séquence d'exception correspondant à l'arrivée de l'information est transparente à l'application. L'appel-moniteur d'acquisition de l'information est exécuté dans cette séquence.
- La séquence d'exception associée à la mise à la disposition de l'information est utilisée pour poster l'évênement

TAM :



VCAM :



Affectation d'une tâche à un correspondant

Avec la méthode d'accès TAM, chaque tâche de l'application SAFRAN est associée à un terminal fixe dont l'index dans la liste des terminaux traités par l'application figure dans la table TCT de la tâche.
Réciproquement, pendant la durée d'exécution de l'application SAFRAN, chacun des terminaux associés ne peut demander d'autres services que ceux de cette application.
Avec la méthode d'accès VCAM, les demandes de connexions sont dynamiques. Un même terminal peut demander une liaison avec plusieurs applications successivement, bien que ces applications soient actives simultanément. Parallèlement, dans une application SAFRAN, aucune tâche n'est attachée à un terminal fixe. Les associations se font dynamiquement.
Une table permet de mémoriser les tâches inactives. Lors d'une demande de connexion, une tâche inactive est affectée au correspondant (dans la limite des places disponibles). Cette affectation est enregistrée dans cette même table. La tâche correspondante est alors réveillée par postage de l'événement bloquant.
Cette affectation est annulée lors d'une déconnexion. La tâche inactive est alors libérée, elle est réutilisable pour une autre demande de connexion ultérieure, pouvant provenir d'un autre correspondant du réseau.

Evolution possible à long terme

Principe

Un des points les plus restrictifs du système SAFRAN actuel est la limitation à un seul du nombre des services d'application résidents simultanément en mémoire.
Il serait peut être utile de développer de nouvelles versions du moniteur permettant un chargement simultané de plusieurs services. Lors de chaque commutation, la séquence de commutation des tâches devrait établir un plan prévisionnel des activations à venir. Pour cela, elle devrait explorer la liste des tâches en aval de la tâche active et effectuer en avance les chargements des services pour le compte des tâches candidates.

Les attentes de chargement seraient plus limitées, les commutations seraient plus rapides. Mais cela se ferait bien sûr au détriment de l'occupation mémoire.

Relocation des services d'application

Si plusieurs services d'application sont résidents simultanément en mémoire, ils ne peuvent être chargés à la même adresse. Cette amélioration du système impose donc la possibilité, pour le logiciel d'application, d'être exécuté à plusieurs emplacements de la mémoire.
Pour qu'un tel algorithme soit plus efficace que l'actuel, il faut que la relocation du logiciel d'application s'effectue sans modification du code.
Puisque la transformation adresse virtuelle --> adresse réelle n'est pas accessible par les services SIRIS 8 standard, cette relocation semble imposer aux programmes de n'utiliser que des références-mémoires basées.
L'utilisation obligée de l'indexation est une restriction très lourde pour l'écriture du logiciel d'application : l'indexation ne serait plus utilisable pour la logique de l'application.
Cette restriction peut être contournée par l'utilisation du mode C de l'IRIS 80. En effet, en permettant l'utilisation de registres de base, le mode rend possible l'écriture aisée de programmes translatables sans altérer les possibilités d'indexation utilisables simultanément :

- si le texte d'un programme a été assemblé et édité pour permettre une exécution avec le texte chargé à l'adresse A et les variables disponibles à l'adresse B, les registres de base contenant alors la valeur 0 ;
- et si le texte et les variables ont été placés pour exécution aux adresses C et D respectivement, les valeurs C-A et D-B doivent être placées respectivement dans les registres de base utilisés pour accéder au texte et aux variables.

Ce noyau de système a été initialement conçu pour une exploitation rationnelle des machines simulées utilisées pour l'enseignement de la programmation.
Son ouverture totale sur SIRIS 8 et son indépendance totale de l'application le destinent à une utilisation générale pour la conception de travaux conversationnels du type "multiterminaux". Ces applications peuvent être des travaux typiquement utilisateur (machines simulées, systèmes de gestion de transactions simples ne nécessitant pas la mise en oeuvre de STRATEGE, ...) ou des travaux du type système (mise au point de programmes par soumission interactive de travaux au moniteur de traitement par trains, interpréteurs de langages évolués, ...).
De plus, sa conception comme tâche SIRIS 8 standard présente des avantages notables d'utilisation :

- Pas de génération de système SIRIS 8 spéciale.
- Possibilité d'avoir plusieurs applications de ce type actives simultanément.
- Utilisation standard des services SIRIS 8 facilitant le suivi des applications lors de changements de version de SIRIS 8.

La mise en oeuvre simple de ce système devrait permettre une meilleure utilisation de l'IRIS 80, la solution simple, mais coûteuse, de passer toutes les tâches conversationnelles sous 'temps partagé' n'étant plus la seule facilement accessible.

A N N E X E A

Interface processeur-système

Appels - système

Lecture au terminal

- Cette fonction effectue une opération d'entrée au terminal. Elle s'accompagne toujours d'une commutation de tâche.
- Elle permet d'introduire un message contenant au maximum le nombre de caractères correspondant à la longueur de ligne définie à la génération. Ce message peut être plus court s'il se termine par un caractère Return.
- La frappe d'un caractère spécial pendant une entrée achève cette opération en anomalie. Le message entré n'est pas significatif.
- L'appel est obtenu par la séquence :

```
CAL2,0      1
GEN,1,31    i,a
```

  + si i=0, a est l'adresse d'octet du buffer
  + si i=1, a est l'adresse d'un mot contenant l'adresse d'octet du buffer.

Ecriture au terminal

- Cette fonction effectue une opération de sortie au terminal. Elle s'accompagne toujours d'une commutation de tâche.
- Elle permet d'envoyer au terminal un message contenant au maximum le nombre de caractères correspondant à la longueur de la ligne définie à la génération.
- L'appel est obtenu par la séquence :

```
CAL2,0      0
GEN,1,31    i,a
GEN,1,31    j,b
```

  + si i=0, a est l'adresse d'octet du buffer
  + si i=1, a est l'adresse d'un mot contenant l'adresse d'octet du buffer.
  + si j=0, b est la longueur du message
  + si j=1, b est l'adresse d'un mot contenant la longueur du message.

Forçage de commutation de tâche

- Cette fonction provoque une commutation, lorsqu'aucune entrée-sortie au terminal n'est nécessaire dans une longue séquence processeur (remplacement du partage du temps)
- L'appel est obtenu par l'instruction :

```
CAL2,0      2
```

Demande de mise en page

- Cette fonction demande au système de rajouter systématiquement les caractères Return et Line-Feed en fin de chaque message envoyé au terminal. (option au lancement de tous les processeurs)
- L'appel est obtenu par l'instruction :

```
CAL2,0      3
```

Suppression de mise en page

- Cette fonction demande au système de ne rien ajouter aux messages envoyés au terminal. La mise en page est alors à la charge du processeur : Un caractère Line-Feed dans le corps d'un message assure le passage à la ligne du terminal.
- L'appel est obtenu par l'instruction :

```
CAL2,0      4
```

Demande d'identification

- Cette fonction permet au processeur d'obtenir l'indicatif de l'usager pour lequel il travaille. Cet indicatif lui est transmis dans un double mot :
  + le 1er mot contient cet indicatif sous la forme de 4 chiffres hexadécimaux
  + le 2e mot contient cet indicatif sous la forme d'un nombre binaire de 8 bits, contenu dans l'octet de fort poids.
- L'appel est obtenu par la séquence :

```
CAL2,0      10
GEN,1,31    i,a
```

  + si i=0, a est l'adresse du double mot où sera rendue l'identification
  + si i=1, a est l'adresse d'un mot contenant l'adresse du double mot où sera rendue l'identification.

Initialisation de sémaphore

- Cette fonction effectue la purge complète d'un sémaphore et initialise la variable associée.
- L'appel est obtenu par la séquence :

```
CAL2,0      8
GEN,1,31    i,a
GEN,1,31    j,b
```

  + si i=0, a est le numéro du sémaphore à initialiser
  + si i=1, a est l'adresse d'un mot contenant le numéro du sémaphore à initialiser.
  + si j=0, b est la valeur à donner à la variable associée
  + si j=1, b est l'adresse d'un mot contenant la valeur à donner à la variable associée.

Opération P sur sémaphore

- Cette fonction effectue l'opération classique P.
  Si le sémaphore est libre, le retour est immédiat
  Si le sémaphore est bloqué, une commutation de tâche est forcée, et la tâche est bloquée jusqu'à déblocage du sémaphore.
- La frappe d'un caractère spécial "Attention" au terminal pendant que la tâche est bloquée provoque le déblocage de la tâche, mais une anomalie est postée.
- L'appel est obtenu par la séquence :

```
CAL2,0      6
GEN,1,31    i,a
```

  + si i=0, a est le numéro du sémaphore désigné
  + si i=1, a est l'adresse d'un mot contenant le numéro du sémaphore désigné.

Opération V sur sémaphore

- Cette fonction effectue l'opération classique V.
  La tâche ne sera jamais bloquée par cette opération, il n'y a jamais de commutation de tâche à cette occasion.
- L'appel est obtenu par la séquence :

```
CAL2,0      7
GEN,1,31    i,a
```

+ si i=0, a est le numéro du sémaphore désigné
+ si i=1, a est l'adresse d'un mot contenant le numéro du sémaphore désigné.

Enchaînement des processeurs

- Cette fonction demande que l'exécution d'un processeur soit immédiatement enchaînée après la fin du processeur en cours.
- Toute demande d'enchaînement écrase la demande précédente d'enchaînement.
- Le processeur à enchaîner est désigné par son nom. Si le processeur existe, la demande est valide. Si le processeur n'existe pas, il n'y aura aucun enchaînement, et une anomalie est postée.
- La séquence suivante provoque cet appel:

```
CAL2,0      11
GEN,1,31    i,a
```

+ si i=0, a est le nom du processeur à enchaîner
+ si i=1, a est l'adresse d'un mot contenant le nom du processeur à enchaîner.

Acquittement de l'anomalie courante

- Cette fonction permet au processeur d'obtenir le code de la dernière anomalie constatée. Ce code lui est transmis sous la forme d'un nombre de 32 bits.
- Le code d'anomalie courant est remis à 0
- L'appel est obtenu par la séquence :

```
CAL2,0      5
GEN,1,31    i,a
```

+ si i=0, a est l'adresse du mot où sera rendu le code d'anomalie
+ si i=1, a est l'adresse d'un mot contenant l'adresse du mot où sera rendu le code d'anomalie.

Fin d'exécution

- Cette fonction retourne le contrôle au système après fin d'exécution d'un processeur. Si une demande d'enchaînement valide a été lancée, le processeur demandé sera alors lancé.

- L'appel est obtenu par l'instruction :

```
CAL2,0      9
```

Accès aux fichiers

Si l'extension d'accès aux fichiers a été générée, les processeurs peuvent accéder aux fichiers standard SIRIS 8 par les appels - moniteur standard, sous réserve qu'ils n' utilisent pas de sections protégées. L'opération d'assignation dynamique est effectuée par l'appel-système supplémentaire :

```
CAL2,0      n
GEN,1,31    i,a
GEN,1,31    j,b
GEN,1,31    k,c
GEN,1,31    l,d
GEN,1,31    m,e
```

- n est le code d'appel-système ; il dépend de l'ordre de déclaration des appels-système à la génération. Si cette extension est employée seule, n sera égal à 12.
- si i=0, a est le numéro du DCB à utiliser (0 ou 1)
  si i=1, a est l'adresse d'un mot contenant le numéro du DCB à utiliser (0 ou 1).
- si j=0, b est l'adresse d'octet du nom du fichier réel (17 caractères)
  si j=1, b est l'adresse d'un mot contenant l'adresse d' octet du nom du fichier réel (17 caractères).
- si k=0, c est le statut du fichier réel (0 : old ; 1 : new)
  si k=1, c est l'adresse d'un mot contenant le statut du fichier réel.
- si l=0, d est l'adresse de la séquence d'anomalie X1, X2 associée au DCB
  si l=1, d est l'adresse d'un mot contenant l'adresse de la séquence d'anomalie X1, X2 associée au DCB.
- si m=0, e est l'adresse d'un mot où le système rendra l'adresse du DCB utilisé
  si m=1, e est l'adresse d'un mot contenant l'adresse d'un mot où le système rendra l'adresse du DCB utilisé.

Les appels-moniteur standard SIRIS 8 utiliseront l'adresse

du DCB ainsi fournie (indirection).

Détection des anomalies

Principe

Les anomalies sont détectées à l'occasion des appels-système. Lorsqu'une anomalie est constatée, un code correspondant à l'anomalie est posté, mais aucune action standard n'est entreprise par le système. Les actions à entreprendre sont à la charge du processeur. Pour ce faire, le processeur peut, lorsqu'il est actif, obtenir à tout moment le code d'anomalie courant par appel-moniteur.

N.B. Certains appels-système ayant une action différente en cas d'anomalie, l'acquittement de l'anomalie courante est quasi obligatoire après ces appels.

Anomalies d'exécution (codes)

code = 0

pas d'anomalie détectée.

code = 1 - code = 2

un caractère Attention 1 ou 2 a été frappé au terminal. Cette anomalie peut être détectée après tout appel moniteur. Si une de ces deux anomalies est détectée à l'occasion d'une lecture au terminal, le message lu n'est pas significatif. Si d'autres anomalies sont détectées en même temps que celles dûes aux caractères Attention, ce code n'est pas transmis.

code = 3

un enchaînement a été demandé avec un processeur inexistant. Cette anomalie ne peut être détectée qu'après un appel de demande d'enchaînement. Lorsqu'une anomalie est détectée, il n'y aura pas d'enchaînement en fin d'exécution du processeur courant.

code = 4

un incident de transfert a été détecté sur la ligne du terminal. Cette anomalie ne peut être détectée qu'après une demande d'accès au terminal (lecture ou écriture)

code = 5

la tâche a été débloquée par désistement. Cette anomalie ne peut être détectée qu'après une opération P sur un sémaphore.

A N N E X E B

Fichier des processeurs

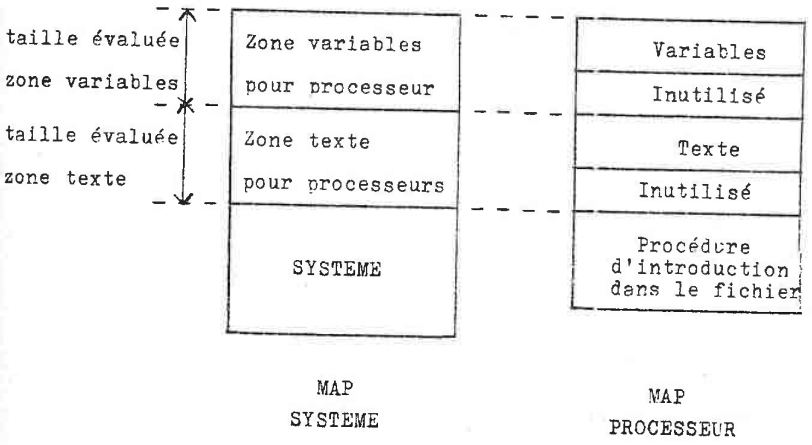
Génération du fichier des processeurs

Principe

Le but de cette opération est de créer le fichier contenant les textes des processeurs (permettant au système d'effectuer les chargements des processeurs).
Ce fichier d'accès direct doit donc contenir les images-mémoire des textes de tous les processeurs, tels qu'ils seront chargés pour exécution.
Les processeurs sont placés séquentiellement dans le fichier, chaque processeur pouvant occuper un ou plusieurs blocs consécutifs, suivant sa taille.

Déroulement de l'opération

Un majorant des tailles des textes des processeurs doit d'abord être évalué, puis un majorant des tailles des zones de variables. Ils permettront de définir la partition mémoire réservée aux processeurs dans le système. L'implantation des processeurs est ainsi fixée.



Les processeurs peuvent alors être introduits un par un.
L'introduction d'un processeur se déroule en trois temps :
- assemblage METASYMBOL du processeur (respectant l'implan-

tation choisie et les conventions propres au système)
- édition du module objet obtenu (processeur et procédure assurant l'introduction dans le fichier)
- chargement en mémoire du module de chargement obtenu et exécution de la fonction assurant la copie sur fichier.

Les renseignements suivants sont fournis au cours du chargement d'un processeur :
- taille exacte de la zone des variables, taille exacte de la zone texte. Ces renseignements sont utilisés lors de la génération du système.
- adresse disque du 1er emplacement libre dans le fichier. En effet, le 1er processeur est toujours introduit à l'adresse 0 (début du fichier). Mais l'emplacement d'introduction du processeur suivant dépend de la taille du précédent : c'est pourquoi chaque opération d'introduction d'un processeur précise à quel emplacement disque doit se faire l'introduction du processeur suivant.
  Ces adresses disques sont également utilisées lors de la génération du système.

Ces renseignements sont disponibles à la fin du listing Metasymbol de compilation :

```
**** ADR DISQUE PROC SUIVANT :
                nnnnnnnn
**** TAILLE TRAVAIL (OCTETS) :
                pppppppp
**** TAILLE TEXTE (OCTETS) :
                qqqqqqqq
```

nnnnnnnn, pppppppp, qqqqqqqq sont les trois nombres fournis, exprimés en base hexadécimale.

Après introduction de tous les processeurs, le fichier est utilisable tel qu'il a été créé.
Il est cependant possible de répéter totalement l'opération, en utilisant une partition mémoire optimale, définie par les bornes supérieures, et non plus par des majorants, des tailles des zones variables et textes de tous les processeurs.

Convention d'écriture des processeurs

Les processeurs doivent être écrits en respectant les conventions suivantes :
- L'implantation choisie pour le texte et les variables doit

être respectée. Des procédures METASYMBOL fournies dans un "système" spécial permettent de la décrire facilement.
- Une séquence spéciale doit être rajoutée au texte du processeur, pour effectuer la copie du texte sur le fichier des processeurs. C'est l'adresse de cette séquence qui doit être fournie comme adresse de début d'exécution, pour l'opération d'introduction sur fichier (directive END).
  Cette séquence est contenue dans une procédure METASYMBOL spécialement fournie.
- Lors de l'exécution du processeur sous système, le contrôle sera initialement passé au 1er mot de la zone texte.
  C'est à cet emplacement que doit être écrite la première instruction du processeur.
- Les appels-système doivent être rédigés conformément à l'interface attendue (voir description). Des procédures METASYMBOL fournies dans un "système" spécial permettent de les rédiger facilement.
- Les séquences-processeur exécutées entre deux appels au système pour accès au terminal ne doivent pas être trop longues. Si tel est le cas, il convient d'ajouter des appels au système judicieusement placés pour forcer la commutation des tâches.
- Des appels à SIRIS 8 peuvent être demandés par les processeurs, sous réserve qu'ils n'entrent pas en conflit avec ceux effectués par le système (procédures TAM notamment), et qu'ils soient conformes à la description de l'implantation choisie (génération de sections protégées).

Mise au point des processeurs

La mise au point des processeurs étant très difficile sous système, il est conseillé, dans la mesure du possible, de mettre au point sous SIRIS 8 TS en respectant les conventions suivantes qui permettront un passage souple :
- séparation du texte et des variables
  (il suffira de rajouter les procédures de description d'implantation).
- la 1ère instruction du processeur sera le 1er mot après les variables.
- utilisation de sous-programmes pour effectuer les accès à la ligne (il suffira de changer quelques appels-système)

- utilisation circonspecte des appels à SIRIS 8 (vérifier que le passage est possible).

Le "système" PROCGEN

Le "système" METASYMBOL PROCGEN contient toutes les procédures nécessaires à l'écriture des processeurs, à savoir :
- les procédures permettant la description de l'implantation mémoire des processeurs
- la procédure contenant la séquence d'introduction du texte dans le fichier des processeurs
- les procédures permettant la génération des appels-système.

Ce "système" référençant lui-même les "systèmes" SIG7 et SIRIS 7 standard, il devra être appelé seul et une seule fois en tête de programme.

Les procédures d'implantation

: INIT (TRAS,i),(TEXS,j),(ADR,n),(BKLS,s)

permet de décrire l'implantation du processeur sur disque et la partition mémoire d'exécution.
Elle doit être appelée une fois et une seule en tête de programme, après l'appel au "système" PROCGEN, et immédiatement avant la zone des variables.
- i est la taille (en octets) de la zone réservée aux variables dans la partition-processeurs du système.
- j est la taille (en octets) de la zone réservée au texte dans la partition-processeurs du système.
- n est l'adresse d'implantation du processeur sur le disque (0 pour le 1er processeur, fournie par le listing de compilation du précédent pour les autres).
- s est la taille de bloc (en octets) du fichier processeurs.

: TRANS

marque la fin de la zone des variables.
Elle doit être appelée une fois et une seule immédiatement après la description des variables et immédiatement avant celle du texte.

: TERM

marque la fin de la zone texte.
Elle doit être appelée une fois et une seule immédiatement après la description du texte.

: PROC

contient la séquence de copie du texte sur le fichier des processeurs.
Elle doit être appelée une fois et une seule immédiatement après l'appel à : TERM
Elle doit toujours avoir une étiquette, qui sera référencée dans la directive END.

Procédures appels-système

Dans la description de la syntaxe des procédures, les accolades précisent que l'une ou l'autre des formes syntaxiques désignées peuvent être utilisées.
exemple :

:ABC (UVW,$\left\{\begin{matrix} ZZ \\ a \end{matrix}\right\}$)

les deux formes suivantes sont possibles :

:ABC (UVW,ZZ)

ou

:ABC (UVW,a)

:IN (REC,$\left\{\begin{matrix} a \\ *b \end{matrix}\right\}$)

permet de générer un appel-système de lecture au terminal.
A utiliser en fonction des besoins.
- a est l'adresse d'octet du tampon (ce dernier est obligatoirement situé en zone variables)
- b est l'adresse d'un mot contenant l'adresse d'octet du tampon (ce dernier est obligatoirement situé en zone variables).

:OUT (REC,$\left\{\begin{matrix} a \\ *b \end{matrix}\right\}$),(TRL,$\left\{\begin{matrix} c \\ *d \end{matrix}\right\}$)

permet de générer un appel-système d'écriture au terminal.
A utiliser en fonction des besoins.
- a est l'adresse d'octet du tampon (ce dernier peut être situé ou non en zone variables)
- b est l'adresse d'un mot contenant l'adresse d'octet

du tampon (ce dernier peut être situé ou non en zone de variables)
- c est la longueur en octets du message à envoyer
- d est l'adresse d'un mot contenant la longueur en octets du message à envoyer.

:IDLE

permet de générer un appel-système de forçage de commutation des tâches. A utiliser en fonction des besoins.

:FON

permet de générer un appel-système de demande de mise en page automatique des messages envoyés au terminal. A utiliser en fonction des besoins.

:FOFF

permet de générer un appel-système de demande de suppression de mise en page automatique des messages envoyés au terminal. A utiliser en fonction des besoins.

:ID (ADR,$\left\{\begin{matrix}a\\ *b\end{matrix}\right\}$)

permet de générer un appel-système de demande d'identification de l'usager. A utiliser en fonction des besoins.
- a est l'adresse d'un double-mot où le système rangera l'identification de l'usager (ce dernier doit être situé en zone variables
- b est l'adresse d'un mot contenant l'adresse d'un double-mot où le système rangera l'identification de l'usager (ce dernier doit être situé en zone variables).

:INITSEM (SEM,$\left\{\begin{matrix}a\\ *b\end{matrix}\right\}$),(VAL,$\left\{\begin{matrix}c\\ *d\end{matrix}\right\}$)

permet de générer un appel-système d'initialisation de sémaphore. A utiliser en fonction des besoins.
- a est le numéro du sémaphore
- b est l'adresse d'un mot contenant le numéro du sémaphore
- c est la valeur à donner à la variable associée au sémaphore
- d est l'adresse d'un mot contenant la valeur à donner à la variable associée au sémaphore.

:P (SEM,$\left\{\begin{matrix}a\\ *b\end{matrix}\right\}$)

permet de générer un appel-système effectuant une opération P sur un sémaphore. A utiliser en fonction des besoins.
- a est le numéro du sémaphore
- b est l'adresse d'un mot contenant le numéro du sémaphore.

:V (SEM,$\left\{\begin{matrix}a\\ *b\end{matrix}\right\}$)

permet de générer un appel-système effectuant une opération V sur un sémaphore. A utiliser en fonction des besoins.
- a est le numéro du sémaphore
- b est l'adresse d'un mot contenant le numéro du sémaphore.

:LINK (NAM ,$\left\{\begin{matrix}a\\ *b\end{matrix}\right\}$)

permet de générer un appel-système initialisant un enchaînement de processeurs. A utiliser en fonction des besoins.
- a est le nom du processeur à enchaîner
- b est l'adresse d'un mot contenant le nom du processeur à enchaîner.

: ACQ (ADR, $\left\{\begin{matrix}a\\ *b\end{matrix}\right\}$)

permet de générer un appel-système d'acquittement d'anomalie. A utiliser en fonction des besoins.
- a est l'adresse d'un mot où le système rangera le code d'anomalie courant (ce dernier est obligatoirement situé en zone variables).
- b est l'adresse d'un mot contenant l'adresse d'un mot où le système rangera le code d'anomalie courant (ce dernier est obligatoirement situé en zone variables).

:EXIT

permet de générer un appel-système de fin d'exécution. A utiliser en fonction des besoins.

Exemple de processeur

Ce programme lit au terminal deux nombres hexadécimaux et

imprime leur somme ; il s'arrête par frappe de A^C.

```
            SYSTEM      PROCGEN
            :INIT       (TRAS,1024),(TEXS,1024),(ADR,0),(BKLS,1024)
BUF         RES,1       80
            BOUND       8
NBR         RES         2
            :TRANS
            :FOFF
            B           DEBUT
*
* SP CONVERSION HEXADECIMAL --> BINAIRE
*
HEXABIN     LI,7        -8
            LI,2        0
HEXABINO    LB,6        8
            CI,6        ' '
            BE          *15
            SLD,8       8
            LI,3        -16
HEXABIN1    CB,6        CARMEX+4,3
            BE          HEXABIN2
            BIR,3       HEXABIN1
            B           ERREUR
HEXABIN2    AI,3        16
            SLS,3       28
            SLD,2       4
            BIR,7       HEXABINO
            B           *15
*
* SP CONVERSION BINAIRE --> HEXADECIMAL
*
BINHEXA     LI,7        -8
BINHEXA1    SLD,8       -8
            SLD,2       -4
            SLS,3       -28
            LB,6        CARMEX,3
            STB,6       8
            BIR,7       BINHEXA1
            B           *15
```



```
*
* SEQUENCE PRINCIPALE
*
DEBUT       :OUT        (REC,BA(M1)),(TRL,11)
            :IN         (REC,BA(BUF))
            :ACQ        (ADR,NBR)
            LW,1        NBR
            CI,1        0
            BNE         STOP
            LD,8        BUF
            BAL,15      HEXABIN
            LW,11       2
            :OUT        (REC,BA(M2)),(TRL,11)
            :IN         (REC,BA(BUF))
            :ACQ        (ADR,NBR)
            LW,1        NBR
            CI,1        0
            BNE         STOP
            LD,8        BUF
            BAL,15      HEXABIN
            AW,2        11
            BAL,15      BINHEXA
            STD,8       NBR
            :OUT        (REC,BA(M3)),(TRL,7)
            :FON
            :OUT        (REC,BA(NBR)),(TRL,8)
            :FOFF
            B           DEBUT
*
* SEQUENCE ERREUR
*
ERREUR      :FON
            :OUT        (REC,BA(M4)),(TRL,7)
            :FOFF
            B           DEBUT
*
* FIN
*
STOP        EXIT
*
M1          TEXT        '1ER NOMBRE?'
M2          TEXT        '2EM NOMBRE?'
M3          TEXT        'SOMME :'
M4          TEXT        'ERREUR.'
```

```
CARMEX          TEXT        '0123456789ABCDEF'
                :TERM
INIT            :PROC
                END         INIT
```

Mise au point des accès aux fichiers

Les processeurs effectuant des accès aux fichiers doivent si possible être mis au point sous SIRIS 8 TS. Ils doivent respecter les conventions standard. De plus, il est conseillé d'utiliser l'indirection pour désigner les DCB's dans les appels-moniteur SIRIS 8 d'accès aux fichiers afin d'éviter la modification de ces appels lors du passage.

Exemple :

- processeur en mise au point sous TS :

```
DCBV            M:DCB       (OPL,'EE'),(BKL,1024),(AM,VS),
 (ABN,X,X1,X2)
                :
DCBUX           DATA        DCBV
                :
ASS             M:ASSIGN    (OPL,'EE'),(NAM,'TRUC      '),
 (STS,OLD),(UNT,OPL,'REF')
                M:OPEN      *DCBUX,I
                M:READ      *DCBUX,(BUF,BA(TP))
                M:CHECK     *DCBUX
                :
```

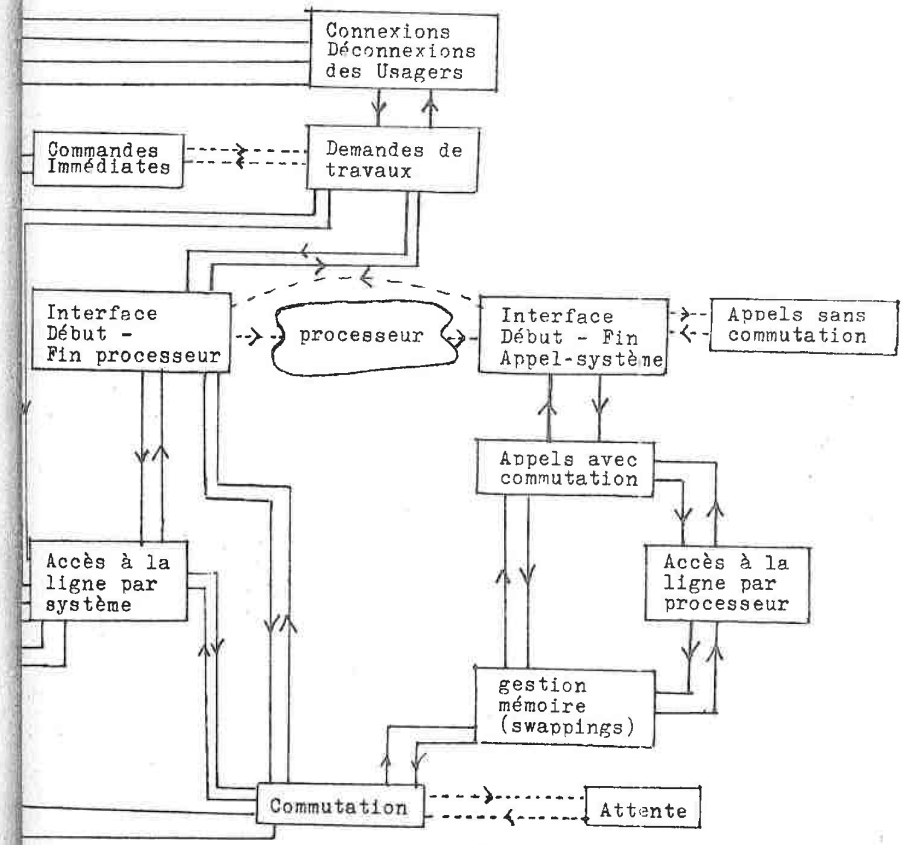
- processeur définitif

```
NOM             TEXT        'TRUC          '
DCBUX           RES         1
                :
ASS             CAL2,0      12
                DATA        0
                DATA        BA(TRUC)
                DATA        0
                DATA        X
                DATA        DCBUX
                M:OPEN      *DCBUX,I
                M:READ      *DCBUX,(BUF,BA(TP))
                M:CHECK     *DCBUX
```


A N N E X E C

Description du système

Description du moniteur



Ce schéma visualise l'enchaînement logique des différentes séquences constituant le moniteur.
Chaque séquence est décrite plus loin.
Chacune d'elles peut constituer :
- une portion de niveau (cas de textes s'enchaînant toujours séquentiellement)

- un niveau (cas des traitements élémentaires communs)
- un ensemble de niveaux (cas de fonctions du même type, appelables dans les mêmes conditions).

Sur le schéma, les traits pleins indiquent des appels avec changement de niveau, les traits pointillés des appels par enchaînement simple, sans changement de niveau.

N.B. : Dans tous les niveaux du système, le registre 12 contient l'adresse de la pile système associée à la tache.

Description du répartiteur

Fonction assurée

Ce niveau assure la commutation des tâches chaque fois qu'une séquence système y fait référence.
Il effectue :
- la perte de contrôle de la tâche usager demandant la commutation.
- le choix et l'activation d'une tâche candidate.
- la mise en attente du système complet si aucune tâche n'est candidate.

Il est constitué de deux traitements séquentiels :
- la commutation proprement dite (module TRTCOMM)
- la mise en attente du système (module TRTWAIT)

Interface à l'appel

Ce niveau peut être appelé à partir des ajouts éventuels (commandes immédiates ou exits connexions-déconnexions) pour forcer la commutation. L'interface suivante doit être respectée :
- Appel par exécution de l'instruction :

  ```
  BAL,15    TRTCOMM
  ```

  (TRTCOMM est une définition externe du système).
- Le point de départ, dans la liste des tâches, de la recherche d'une tâche candidate doit être désigné comme suit :
  l'adresse de la table du contexte (TCT) de la tâche précédent la première tâche consultée, doit être contenue dans la table du contexte commun (TRG) à l'entrée TCHACT.
  lors d'une activation, la variable TCHACT désigne la tâche activée. Si l'ordre de consultation de la liste des tâches ne doit pas être modifié, il n'y a pas lieu d'altérer cette variable avant la désactivation.
- Le type de désactivation doit être précisé dans la table du contexte de la tâche à désactiver, à l'entrée INDID
  X'FF' : tâche immédiatement candidate à la réactivation
  0 : réactivation conditionnée par l'accomplissement de l'évènement désigné par l'entrée ECBD de la table du contexte de la tâche.
- L'appel de ce niveau nécessite une entrée libre au sommet de la pile.

Interface au retour

- Retour derrière l'instruction d'appel.
- La variable TCHACT de la table du contexte commun (TRG) et le registre 1 désignent la table du contexte de la tâche activée (TCT).

Commutation avec gestion mémoire

Fonction assurée

Ce niveau assure la commutation des tâches chaque fois qu'une désactivation est demandée par une séquence système à la suite d'une requête processeur (la mémoire doit être sauvegardée).
Il effectue :
- les swappings de sauvegarde avant désactivation
- la commutation (appel au répartiteur)
- les swappings de restauration après réactivation.

Il est constituée du module TRTSWAP.

Interface d'appel

Ce niveau peut être appelé à partir des ajouts éventuels (appels-système) pour forcer la commutation. L'interface suivante doit être respectée :

- Appel par exécution de l'instruction

  BAL,15 TRTSWAP

  (TRTSWAP est une définition externe du système).
- Le point de départ de la recherche d'une tâche candidate est soumis aux mêmes regles que pour un appel au répartiteur.

  Le type de désactivation doit également être précisé de la même manière.
- L'appel de ce niveau nécessite deux entrées libres au sommet de la pile.

Interface retour

Elle est soumise aux mêmes règles que pour un appel au répartiteur.

Accès à la ligne hors processeur

Fonctions assurées

C'est un ensemble de deux niveaux similaires assurant les accès à la ligne associée à la tâche, demandés par le sytème lorsqu'aucun processeur n'est actif pour cette tâche.
Ils n'entraînent donc aucune gestion mémoire.
Un des niveaux assure l'opération d'entrée, l'autre la sortie.
Ils effectuent :

- le lancement et le contrôle de l'opération d'accès
- la désactivation de la tâche pendant la durée de l'opération (la commutation est demandée par un appel au répartiteur).

Le niveau effectuant l'entrée est composé du module TRTGSYS.
Le niveau effectuant la sortie est composé du module TRTPSYS.

Interface à l'appel du niveau : lecture système

Ce niveau peut être appelé à partir des ajouts éventuels (commandes immédiates ou exits connexions-déconnexions) pour effectuer des entrées sur la ligne associée. L'interface suivante doit être respectée :

- Appel par exécution de l'instruction :

  BAL,15 TRTGSYS

  (TRTGSYS est une définition externe du système).
- Lors de l'appel :
  l'adresse d'octet du buffer doit être dans le registre 4
  l'adresse d'octet de la chaîne de caractères préfixes doit être dans le registre 5
  la longueur de la chaîne de caractères préfixes doit être dans le registre 3.

L'appel de ce niveau nécessite deux entrées libres au sommet de la pile.

Interface au retour

- Retour derrière l'instruction d'appel.
- La réponse du terminal se trouve dans le buffer choisi.
- le registre 0 contient le code d'achèvement de l'opération physique lancée par le niveau.

- La variable RETURN de la table du contexte commun (TRG) contient le nombre d'octets constituant la réponse.

Interface à l'appel du niveau : écriture système

Ce niveau peut être appelé à partir des ajouts éventuels (commandes immédiates ou exits connexions-déconnexions) pour effectuer des sorties sur la ligne associée. L'interface suivante doit être respectée :

- Appel par exécution de l'instruction :

  BAL,15 TRTPSYS

  (TRTPSYS est une définition externe du système).
- Lors de l'appel :
  l'adresse d'octet du buffer (message à écrire) doit être dans le registre 4
  la longueur de la chaîne de caractères constituant le message doit être dans le registre 3.
- L'appel de ce niveau nécessite deux entrées libres au sommet de la pile.

Interface au retour

- Retour derrière l'instruction d'appel.
- Le registre 0 contient le code d'achèvement de l'opération physique lancée par le niveau.

Accès à la ligne demandés par un processeur

Fonctions assurées

C'est un ensemble de deux niveaux similaires assurant les accès à la ligne associée à la tâche, demandés par le système pour l'exécution d'une requête-processeur.
Cette opération demande donc une gestion mémoire.
Un des niveaux assure l'opération d'entrée, l'autre la sortie.
Ils effectuent :

- le lancement et le contrôle de l'opération d'accès
- la désactivation de la tache, et la gestion mémoire associée pendant la durée de l'opération.

(la commutation est demandée par un appel au niveau de commutation avec gestion mémoire).

Le niveau effectuant l'entrée est composé du module TRTGUSE.
Le niveau effectuant la sortie est composé du module TRTPUSE.

Interface à l'appel du niveau : lecture par processeur

Ce niveau peut être appelé à partir des ajouts éventuels (appels-système) pour effectuer des entrées sur la ligne associée. L'interface suivante doit être respectée :

- appel par exécution de l'instruction

  BAL,15 TRTGUSE

  (TRTGUSE est une définition externe du système).
- les registres 3, 4, 5 doivent être positionnés lors de l'appel de ce niveau comme lors de l'appel du niveau : lecture système.
- l'appel de ce niveau nécessite 3 entrées libres au sommet de la pile.

Interface au retour

Elle est identique à l'interface au retour du niveau lecture système.

Interface à l'appel du niveau : écriture par processeur

Ce niveau peut être appelé à partir des ajouts éventuels (appels-système) pour effectuer des sorties sur la ligne

associée.
L'interface suivante doit être respectée :

- appel par exécution de l'instruction :

  BAL,15 TRTPUSE

  (TRTPUSE est une définition externe du système).
- les registres 3 et 4 doivent être positionnés lors de l'appel de ce niveau comme lors de l'appel du niveau : écriture système.
- l'appel de ce niveau nécessite trois entrées libres au sommet de la pile.

Interface au retour

Elle est identique à l'interface au retour du niveau écriture système.

Gestion des connexions

Fonction assurée

Ce niveau assure la réception et le traitement des demandes de connexion et de déconnexion des usagers.
C'est le premier niveau appelé lors du lancement d'une tâche usager.
Il effectue :

- la réception des demandes de connexion
- l'acceptation ou le refus de la demande
- le lancement du niveau : demande de travaux
- la mise au repos de la tâche avec attente des demandes de connexion lorsqu'une demande de déconnexion a été détectée par le niveau : demande de travaux.
- un dialogue avec l'opérateur central pour l'avertir de la charge du système.

Il est constitué par le module RTLOG.

Interface avec le reste du système

Ce niveau est le niveau initial : son exécution est lancée au niveau 0 de la pile (pile vide). Il n'est appelé par aucun module.
Il appelle les niveaux :

- répartiteur sans gestion mémoire : pour mise au repos de la tâche
- écriture système : pour les messages envoyés au terminal associé lors de la connexion et de la déconnexion
- demande de travaux : pour le lancement effectif de la tâche.

Interface avec les séquences usager

Des séquences usager peuvent être rajoutées à ce niveau. Elles seront exécutées lors de chaque connexion après les opérations standard et lors de chaque déconnexion, avant les opérations standard.
Elles sont soumises aux règles suivantes :

- leur texte est considéré comme une portion du niveau : gestion des connexions. Elles sont donc lancées sur le niveau 0 de la pile, par simple branchement. Leur texte peut être ou non découpé en plusieurs niveaux.
- au lancement de la séquence, toutes les tables du système sont accessibles. En particulier, la variable TCHACT de la table du contexte commun (TRG) et le registre 1 désignent la table du contexte de la tâche active (TCT).
- leur texte peut appeler d'autres niveaux pour exécuter des fonctions de base, par exemple :
  Commutation : appel du niveau : répartiteur (sans gestion mémoire)
  Accès à la ligne : appel des niveaux : lecture ou écriture système.
- le retour en fin d'exécution se fait par
  B RETLI
  pour la fin de la séquence à la connexion
  B RETLO
  pour la fin de la séquence à la déconnexion
  (RETLI et RETLO sont deux définitions externes du système).
- au retour, le registre 1 doit désigner la table du contexte de la tâche active (TCT).

Demande de travaux

Fonctions assurées

Ce niveau assure le dialogue avec l'usager du terminal, la réception et le traitement des requêtes transmises par ce dernier.
Il effectue :

- l'envoi au terminal des messages déposés dans la boîte aux lettres de la tâche active
- la réception des requêtes de l'usager
- l'analyse et le refus éventuel de ces requêtes
- l'exécution de la commande immédiate, si l'une d'elles est demandée
- les modifications de la liste des tâches et l'appel du niveau : exécution des processeurs, si l'un d'eux est demandé.

Il est constitué des traitements suivants :

- dialogue avec l'usager (module TRTJOB)
- modification de la liste des taches (avant et après exécution d'un processeur (module TRTACCS)
- commandes immédiates.

Interface avec le reste du système

Ce niveau est appelé par le niveau : gestion des connexions.
Son exécution est lancée au niveau 1 de la pile.
Il appelle les niveaux :

- Lecture et Ecriture système : pour le dialogue avec l'usager du terminal
- Exécution des processeurs : pour le lancement, l'exécution et le retour d'un processeur.

Interface avec les commandes immédiates

Des commandes immédiates peuvent être ajoutées au système.
Elles sont soumises aux règles suivantes :

- leur texte est considéré comme une portion du niveau : Demande de travaux. Elles sont donc lancées sur le niveau 1 de la pile, par simple branchement.
  Le texte de la commande peut être découpé ou non en plusieurs niveaux.

- au lancement de la commande, toutes les tables du système sont accessibles. En particulier, la variable TCHACT de la table du contexte commun (TRG) et le registre 1 désignent la table du contexte de la tâche active (TCT)
- le texte de la commande peut appeler d'autres niveaux pour exécuter des fonctions de base. Par exemple :
  Commutation : appel du niveau : répartiteur (sans gestion mémoire)
  Accès à la ligne : appel des niveaux : lecture ou écriture système
- le retour en fin d'exécution de commande se fait par exécution de l'instruction :

```
B          TRTPROC
```

(TRTPROC est une définition externe du système)

- au retour, le registre 1 doit contenir l'adresse de la table du contexte de la tâche active (TCT).

Exécution des processeurs

Fonctions assurées

Ce niveau assure les fonctions d'interface processeur-système au lancement, à l'exécution, et au retour du processeur.
Il effectue :

- les swappings d'initialisation
- l'exécution du processeur
- les swappings de sauvegarde en fin d'exécution
- la réception des appels-système
- l'exécution des appels-système sans commutation
- le contrôle de la surveillance parallèle du terminal
- l'appel des niveaux : appels-système (avec commutation)

Il est constitué des traitements suivants :

- swappings initiaux et finaux (module TRTEXEC)
- réception des appels système (module TRTCAL)
- appels système sans commutation (modules TRTFMT, TRTINT, TRTID, TRTVASY)
- texte des processeurs .

Interface avec le reste du système

Ce niveau est appelé par le niveau : demande de travaux.
Il est lancé au niveau 2 de la pile.
Le texte de tout appel-système est lancé au niveau 2 de la pile, par simple branchement. Si l'appel-système donne lieu à une commutation, il convient de sauver sur la pile l'adresse de l'appel moniteur dans le texte du processeur, afin de pouvoir reprendre l'exécution au bon endroit. Une telle séquence constitue alors un niveau sans en avoir l'appel standard.

Interface avec les appels-système

Des appels-système peuvent être rajoutés au texte standard ; ils doivent respecter les règles suivantes :

- au lancement, toutes les tables-système sont accessibles. En particulier, la variable TCHACT de la table du contexte commun (TRG) et le registre 1 désignent la table du contexte de la tâche active (TCT) ;
  la variable BRCHADR de la table du contexte commun (TRG)

contient l'adresse de l'appel-moniteur responsable.

- le texte du traitement peut appeler des niveaux pour exécuter des fonctions de base :
  commutation par appel du niveau : répartiteur avec gestion mémoire
  accès à la ligne par appel des niveaux : lecture et écriture processeur
- l'adresse de reprise doit être placée dans la variable BRCHADR de la table du contexte commun (TRG) avant le retour (l'adresse de reprise est la 1ère instruction après les paramètres de l'appel-moniteur)
- le retour au niveau : exécution des processeurs doit se faire par exécution de l'instruction :

```
B          TRTFCAL
```

(TRTFCAL est une définition externe du système)

- au retour, le registre 1 doit désigner la table du contexte de la tâche active (TCT).

Lancement du système

Une séquence d'initialisation est exécutée avant le démarrage réel du système. Cette séquence effectue les opérations standard et lance réellement le système.
Une séquence utilisateur peut être ajoutée à la séquence standard. Elle sera exécutée immédiatement avant le lancement réel du système, et permet d'initialiser les fonctions ajoutées.

Elle doit respecter les conventions suivantes :

- au lancement, toutes les tables système sont accessibles. En particulier, la variable TCHACT de la table du contexte commun (TRG), et le registre 1 contiennent l'adresse de la table du contexte de la première tâche (TCT)
- le système n'étant pas encore lancé lors de l'exécution de la séquence, les fonctions de base du système ne doivent pas être accédées.
- la séquence assure le retour au système par exécution de l'instruction :

  B RETRET

  (RETRET est une définition externe du système).

Codes d'achèvement des opérations d'accès ligne

X'00'	:	Attention 1 en entrée
X'01'	:	Attention 2 en entrée
X'02'	:	ETX en entrée
X'03'	:	Délai de réception dépassé en entrée
X'04'	:	Purge par M:MDFLST (surveillance seulement)
X'05'	:	incident en ligne en entrée ou sortie
X'06'	:	fin normale en entrée ou sortie

Tables système

Table du contexte commun, ou table des renseignements généraux (TRG)

Cette table contient tous les renseignements communs à toutes les tâches du système.Toutes ses entrées sont des définitions externes du système.
Elle a la structure suivante :

0	: TCHACT	7	: NBSEM
1	: TCHIN	8	: LTNOM
2	: POPRO	9	: NTERAC
3	: NBPRO	10	: RETURN
4	: LNGBU	11	: BRCHADR
5	: LSTSWP	12	: TEMSTOP
6	: NBCNS		

- TCHACT
  désigne généralement par son adresse, la table du contexte de la tâche active. Permet de désigner le point de départ de la recherche de tâche à activer après commutation. (voir commutation).
- TCHIN
  désigne par son adresse la table du contexte de la dernière tâche hors processeur dans la liste des TCT.
- POPRO
  pointeur dans la table de description des processeurs (TDP) Désigne le processeur actuellement chargé dans la partition processeurs du système.
- NBPRO
  contient le nombre de processeurs utilisables.
- LNGBU
  contient la longueur de ligne du terminal.
- LSTSWP
  contient l'adresse dans le fichier swapping du dernier bloc utilisable par le swapping.
- NBCNS
  contient le nombre de terminaux gérés par le système.
- NBSEM
  contient le nombre de sémaphores disponibles.

- LTNOM
  contient le nombre de commandes immédiates utilisables.
- NTERAC
  contient le nombre de terminaux connectés (donc de tâches éveillées).
- RETURN
  à l'issue d'un accès à la ligne en entrée, contient le nombre d'octets frappés au terminal.
- BRCHADR
  pendant le traitement d'un appel-système, contient l'adresse de l'instruction responsable de l'appel (CAL2).
- TEMSTOP
  initialisé à 0 au lancement du système, devient non nul par demande d'arrêt opérateur.

Table du contexte de tâche (TCT)

Ces tables existent dans le système à raison de une par tâche déclarée à la génération. Chacune contient toutes les informations constituant le contexte de la tâche associée.
Ces tables sont chaînées entre elles. Elles ne peuvent être atteintes que grâce à la variable TCHACT de la table TRG et à leur chaînage.
Elles ont la structure suivante :

0	: CHAVD	10	: CONSD
1	: CHAMD	11	: ABUFD
2	: INDID	12	: ADRTD
3	: ACTID	13	: ENCHD
4	: ECBD	14	: PBOXD
5	: ECBXD	15	: USERD
6	: REGID	16	: SEMAD
7	: SAVED	17	: AUX
8	: PPROD	18	: BIDND
9	: SWAPD		

- CHAVD
  chaînage aval.
- CHAMD
  chaînage amont.
- INDID
  sert à la commutation des tâches :
  Si 0 : la tache ne sera candidate à la réactivation que quand l'événement pointé par ECBD sera posté

Si X'FF' : la tâche est immédiatement candidate à la réactivation.

- ACTID

Si 0 : aucun processeur n'est actif pour cette tâche
Si X'FF' : un processeur est actif pour cette tâche.

- ECBD

contient l'adresse de l'évênement bloquant la réactivation de la tâche.

- ECBXD

contient l'adresse de l'évênement associé à la surveillance parallèle.

- REGID

contient l'adresse de la zone de 16 mots servant à la sauvegarde des registres processeur sur appel-système ou en fin d'exécution.

- SAVED

contient l'adresse de la pile-système.

- PPROD

pointeur dans la table de description des processeurs (TDP). Si un processeur est actif pour la tâche (voir ACTID), désigne ce processeur.

- SWAPD

contient l'adresse dans le fichier de swapping, du premier bloc utilisable pour la sauvegarde de la zone variable associée à la tâche.

- CONSD

contient dans l'octet de fort poids l'index du terminal associé à la tâche dans la "liste" TAM .

- ABUFD

contient l'adresse d'octet du tampon utilisé pour les accès à la ligne.

- ADRTD

contient le code d'anomalie courant.

- ENCHD

si l'octet de fort poids est nul, aucun enchaînement de processeurs n'est demandé.
si l'octet de fort poids vaut X'FF', le reste du mot contient un pointeur dans la table TDP vers le processeur demandé en enchaînement.

- PBOXD

l'octet de fort poids contient l'état de la boîte aux lettres.

Le reste du mot contient l'adresse de la boîte aux lettres associée à la tâche.
valeurs possibles de l'octet de fort poids :

0 : boite vide
X'80' : un message est en cours d'introduction
X'FF' : un message est disponible.

- USERD

contient l'indicatif de la tâche usager (4 caractères EBCDIC représentant un nombre hexadécimal de 4 chiffres).

- SEMAD

lorsque la tâche est bloquée sur un sémaphore, contient l'adresse du sémaphore bloquant.

- AUX

zone de travail de la commande immédiate 'POS'.

- BIDND

contient l'adresse de la zone propre ajoutée (si l'extension a été déclarée à la génération - procédure : zone).

Table de description des processeurs (TDP)

C'est un ensemble de 6 tables contenant chacune autant d'entrées qu'il y a de processeurs déclarés dans le système.
L'ensemble des entrées de même rang dans ces six tables constitue la description d'un processeur.
Les adresses de chacune de ces six tables sont des définitions externes du système.
Exemple de structure pour trois processeurs :

0....TPRONOM	9....TPROTTX
1	10
2	11
3....TPROADR	12....TPRONBR
4	13
5	14
6....TPROTSW	15....TPROPNT
7	16
8	17

- TPRONOM

chaque entrée contient le nom du processeur associé.

- TPROADR

chaque entrée contient l'adresse, dans le fichier des processeurs, du premier bloc contenant le texte du processeur associé.

- TPROTSW
  chaque entrée contient la taille, en octets, de la zone des variables du processeur associé.
- TPROTTX
  chaque entrée contient la taille, en octets, de la zone texte du processeur associé.
- TPRONBR
  chaque entrée contient le nombre de tâches-usager actuellement sous contrôle du processeur associé.
- TPROPNT
  chaque entrée contient l'adresse de la table du contexte (TCT) de la dernière tâche sous contrôle du processeur associé, dans la liste des TCT.

Sémaphores

La table des adresses des sémaphores contient les adresses des descripteurs de tous les sémaphores déclarés dans le système. Elle a autant d'entrées qu'il y a de sémaphores déclarés : chacune contient l'adresse du descripteur du sémaphore associé. L'adresse de cette table (ADSEM) est une définition externe du système.
Les descripteurs de sémaphores contiennent chacun les informations associées à un sémaphore. Ils existent dans le système à raison de un par sémaphore déclaré à la génération. Ils sont repérables grâce à la table des adresses des sémaphores.
Ils ont la structure suivante :

0 : SMAPH
1 : SMLIB
2 : SMCAN
3 : SMFIL

- SMAPH
  contient la variable associée au sémaphore
- SMFIL
  contient l'adresse de la file d'attente associée
- SMCAN
  contient l'index du premier candidat dans la file
- SMLIB
  contient l'index de la première position libre dans la file.

La file d'attente est un tableau comptant autant d'entrées qu'il y a de tâches déclarées à la génération. Cette file est gérée circulairement. Les tâches mises en attente dans cette file sont

désignées par l'adresse de la table du contexte (TCT) associée.

Boîtes aux lettres

Ces tables contiennent les messages destinés aux terminaux. Elles existent dans le système à raison de une par tâche déclarée à la génération. Chaque table est repérable grâce à un pointeur contenu dans la table du contexte de la tâche associée.
Les messages contenus dans ces boîtes ont l'aspect suivant :

- indicatif de l'expéditeur (4 octets)
- longueur du message (4 octets)
- texte du message (au maximum : longueur de ligne du terminal).

Pile système

Ces tables existent dans le système à raison de une par tâche déclarée à la génération. Chaque table est repérable par un pointeur contenu dans la table du contexte de la tâche associée (TCT). Leur structure est celle d'une pile standard IRIS80.

Table des services

Cette table est unique pour le système. Elle contient autant d'entrées qu'il y a d'appels-système utilisables ; chaque entrée contient une instruction de branchement inconditionnel vers le traitement de cet appel-système. L'index d'une entrée dans cette table a la même valeur que le code de l'appel-système correspondant.
L'adresse de cette table est une définition externe du système (TRTCALT).

Table des commandes immédiates

C'est un ensemble de deux tables contenant chacune autant d'entrées qu'il y a de commandes immédiates utilisables ; l'ensemble des entrées de même rang de ces deux tables constitue la description d'une commande. Les adresses de ces deux tables sont des définitions externes du système.

- TNOMCM
  chaque entrée contient le nom de la commande associée.
- TADRCM
  chaque entrée contient l'adresse de la séquence de traitement de la commande associée.

A N N E X E D

Génération du système

Génération du système standard

Les "systèmes" METASYMBOL

Deux "systèmes" sont utilisés pour effectuer cette opération :

- le premier (SYSGEN) contient les procédures permettant de fixer les valeurs des paramètres, et de vérifier leur cohérence.
- le deuxième (SYSTAB) contient les directives effectuant la construction proprement dite des tables variables.

La description d'une version de système se compose donc de trois parties logiques :

- création des procédures par appel du "système" SYSGEN
- description des caractéristiques souhaitées par l'appel des procédures contenues dans SYSGEN. L'appel peut se faire dans n'importe quel ordre.
- création des tables par appel du "système" SYSTAB.

exemple :

```
SYSTEM          SYSGEN
   :
procédures décrivant les caractéristiques désirées
   :
SYSTEM          SYSTAB
END
```

Les procédures du SYSGEN

:TERMIN (NBR,n),(REL,p)

permet de décrire le réseau de terminaux.
Elle doit être appelée 1 fois et une seule.
- n indique le nombre de terminaux connectables
- p indique la longueur d'une ligne (en caractères) si le nombre de caractères sur la ligne n'est pas un multiple de 4, prendre le multiple de 4 immédiatement supérieur.

:MEMO (TRAV,n),(TEXT,p)

permet de décrire l'espace mémoire réservé pour l'exécution des processeurs :
Elle doit être appelée 1 fois et une seule.
- n indique la taille de la zone des variables en octets

(la plus grande de celles des processeurs)
- p indique la taille de la zone de texte, en octets (la plus grande de celles des processeurs).

:SWAPF (TVRL,n),(TXRL,p)

permet de décrire les tailles de blocs des fichiers de processeurs et de swapping.
- n est la taille des blocs du fichier swapping, en octets
- p est la taille des blocs du fichier processeurs, en octets (voir création du fichier processeurs).

:SEMA (VAL,n)

permet de déclarer les sémaphores utilisés.
Elle doit être appelée autant de fois qu'il doit exister de sémaphores.
- n est la valeur initiale de la variable associée.
- le numéro du sémaphore est fixé par l'ordre d'appel des procédures :SEMA :
  le 1er appel crée le sémaphore 0
  le 2° appel crée le sémaphore 1...

:PROCES (NOM,t),(TTRA,n),(TTEX,p),(ADRD,r)

permet de décrire les processeurs associés.
Elle doit être appelée autant de fois qu'il existe de processeurs associés.
- t est le nom du processeur (4 caractères EBCDIC)
- n est la taille de la zone des variables, en octets
- p est la taille du texte du processeur, en octets
- r est l'adresse du 1er bloc contenant le texte dans le fichier des processeurs.

n, p, r, sont fournis par la création du fichier processeurs.

:APNAM (NAM,t)

permet de décrire le nom de l'application, celui sous lequel le système sera connu de l'opérateur central (utile si plusieurs versions du système sont actives à un instant donné).
Elle doit être appelée une fois et une seule.
t est le nom de l'application (4 caractères EBCDIC).

Exemple de génération standard

Caractéristiques du système :
- nom de l'application APPL
- 4 terminaux à 72 caractères par ligne
- 3 processeurs dont les caractéristiques sont les suivantes :

Nom	taille variables	taille texte	Adresse fichier
TOTO	1000	1760	0
TRUC	2250	838	2
CHOS	1700	2200	3

- 2 sémaphores initialisés à
  2 , 1
- le fichier swapping, le fichier processeurs seront placés sur des disques de taille secteur : 1024 octets.

Ce système se décrit de la façon suivante :

```
        SYSTEM    SYSGEN
        :TERMIN   (NBR,4),(REL,72)
        :PROCES   (NOM,'TOTO'),(TTRA,1000),;
(TTEX,1760),(ADRD,0)
        :PROCES   (NOM,'TRUC'),(TTRA,2250),;
(TTEX,838),(ADRD,2)
        :PROCES   (NOM,'CHOS'),(TTRA,1700),;
(TTEX,2200),(ADRD,3)
        :SEMA     (VAL,2)
        :SEMA     (VAL,1)
        :MEMO     (TRAV,2250),(TEXT,2200)
        :SWAPF    (TVRL,1024),(TXRL,1024)
        :APNAM    (NAM,'APPL')
        SYSTEM    SYSTAB
        END
```

Génération d'un système étendu

Principe

Le principe est le même que pour une génération standard. Le "système" SYSTAB doit cependant créer des informations supplémentaires pour faire le lien entre le texte du système et les ajouts compilés séparément. Ces informations contiennent entre autres les adresses de ces séquences, qui doivent être déclarées externes à la génération. Leur déclaration est assurée par leur mention dans des procédures spéciales du système SYSGEN.
Les identificateurs utilisés pour ces adresses peuvent être théoriquement tout symbole admis par l'assembleur METASYMBOL, mais les symboles déjà utilisés par la génération ne doivent pas être réutilisés.
Un moyen simple d'éviter toute double déclaration est de les faire toutes commencer par la chaîne de caractères U≠, non employée dans les symboles du système.

Les procédures de déclaration d'extensions

:CAL (ADR,n)

permet de déclarer un appel-système supplémentaire. Elle doit être déclarée autant de fois qu'il y a de séquences ajoutées.
- n est l'adresse du début du traitement associé. Cette adresse doit être déclarée référence externe.
- L'ordre d'appel des procédures :CAL fixe l'ordre d'introduction des traitements supplémentaires dans la table.
  Ainsi le traitement décrit par le premier appel de la procédure :CAL est exécuté lors de la rencontre d'une instruction CAL2,0 12
  Celui du deuxième appel par la rencontre de l'instruction CAL2,0 13 etc...

:COM (NOM,t),(ADR,n)

permet de déclarer une commande immédiate supplémentaire. Elle doit être appelée autant de fois qu'il y a de commandes ajoutées.
- t est le nom de la commande (4 caractères EBCDIC)
- n est l'adresse du début du traitement associé. Cette adresse doit être déclarée référence externe.

:EXIT (ADR,n)

permet de déclarer une séquence usager à exécuter au lancement du système. Elle ne doit être appelée qu'une fois.
- n est l'adresse du début du traitement associé. Cette adresse doit être déclarée référence externe.

:EXLI (ADR,n)

permet de déclarer une séquence usager à exécuter lors de la connexion d'un usager. Elle ne doit être appelée qu'une fois.
- n est l'adresse du début du traitement associé. Cette adresse doit être déclarée référence externe.

:EXLO (ADR,n)

permet de déclarer une séquence usager à exécuter lors de la déconnexion d'un usager. Elle ne doit être appelée qu'une fois.
- n est l'adresse du début du traitement associé. Cette adresse doit être déclarée référence externe.

:ZONE (TAIL,n)

permet de déclarer une zone de travail propre à chaque tâche (faisant partie du contexte de la tache). Elle ne doit être appelée qu'une fois.
- n est la taille, en mots, de la zone de travail.

:PILE (SUPL,n)

permet de demander un allongement de la pile système associée à chaque tâche (implicitement 8 mots). Elle ne doit être appelée qu'une fois.
- n est le nombre de double-mots ajoutés à la pile de chaque tâche.

Exemple de génération étendue

Les caractéristiques du système à générer sont les mêmes que celles du système généré dans l'exemple de génération standard, avec en plus :

- 1 processeur de nom MACH
  de taille de zone de variables : 1500
  de taille de zone texte : 1200
  situé dans le fichier processeur à l'adresse : 6
- ce processeur utilise le traitement commençant à l'adresse U≠TRT1 par l'appel moniteur CAL2,0 12 et le traitement commençant à l'adresse U≠TRT2 par l'appel moniteur CAL2,0 13
- ces traitements utilisent une zone de travail propre de 18 mots.
- une commande immédiate de nom 'KJLM' doit être ajoutée, dont le traitement commence à l'adresse U≠COMM
- la séquence commençant à l'adresse U≠INIT doit être exécutée au lancement du système
- la séquence commençant à l'adresse U≠LOGIN doit être exécutée à la connexion d'un usager.
- la séquence commençant à l'adresse U≠LOGOUT doit être exécutée à la déconnexion d'un usager
- la pile système doit être allongée de 5 mots.

Ce système se décrit de la façon suivante :

```
          SYSTEM      SYSGEN
          :TERMIN     (NBR,4),(REL,72)
          :PROCES     (NOM,'TOTO'),(TTRA,1000),;
(TTEX,1760),(ADRD,0)
          :PROCES     (NOM,'TRUC'),(TTRA,2250),;
(TTEX,838),(ADRD,2)
          :PROCES     (NOM,'CHOS'),(TTRA,1700),;
(TTEX,2200),(ADRD,3)
          :PROCES     (NOM,'MACH'),(TTRA,1500),;
(TTEX,1200),(ADRD,6)
          :SEMA       (VAL,2)
          :SEMA       (VAL,1)
          :MEMO       (TRAV,2250),(TEXT,2200)
          :SWAPF      (TVRL,1024),(TXRL,1024)
```



```
          REF         U≠TRT1
          REF         U≠TRT2
          REF         U≠COMM
          REF         U≠INIT
          REF         U≠LOGIN
          REF         U≠LOGOUT
          :CAL        (ADR,U≠TRT1)
          :CAL        (ADR,U≠TRT2)
          :COMM       (ADR,U≠COMM)
          :EXIT       (ADR,U≠INIT)
          :EXLI       (ADR,U≠LOGIN)
          :EXLO       (ADR,U≠LOGOUT)
          :ZONE       (TAIL,18)
          :PILE       (SUPL,3)
          SYSTEM      SYSTAB
          END
```

Génération d'un système avec accès aux fichiers

Conventions suivies par l'extension

- l'extension est composée :
  + d'une séquence exécutée sur appel-système, débutant à l'adresse U≠CAL
  + d'une séquence exécutée au lancement du système, débutant à l'adresse U≠EXIT
- Elle utilise les 2 premiers mots de la zone propre ajoutée
- Le retour de la séquence exécutée au lancement se fait par

```
B          RETRET
```

Génération utilisant cette seule extension

Les directives suivantes doivent être ajoutées à la génération standard :

```
REF        U≠CAL
REF        U≠EXIT
:CAL       (ADR,U≠CAL)
:EXIT      (ADR,U≠EXIT)
:ZONE      (TAIL,2)
```

Le service ajouté peut alors être appelé par un processeur par exécution de l'instruction :

```
CAL2,0     12
```

Conventions imposées à d'éventuelles extensions supplémentaires

- séquences appels-système
  aucune convention supplémentaire
- séquences commandes immédiates
  aucune convention supplémentaire
- séquences à la connexion et à la déconnexion
  aucune convention supplémentaire
- séquence au lancement
  la séquence au lancement initialisant l'extension d'accès aux fichiers, effectuant un retour au système en fin d'exécution, toute séquence supplémentaire devra être exécutée avant. Elle devra donc

  + être elle même déclarée comme seule séquence au lancement (procèdure :EXIT à la génération)
  + assurer le lancement de la séquence d'initialisation des accès-fichier (se terminer par

```
B          U≠EXIT
```

  avec

```
REF        U≠EXIT     )
```

Les codes d'appel des services ajoutés dépendent dans ce cas de l'ordre de déclaration des séquences appel-système à la génération (procédures :CAL).

Exemple de conception d'extension

Principe de l'exemple

Tenir à jour un fichier du type "comptabilité" où seront enregistrés des articles mémorisant les évènements suivants et leur date (fournie par appel moniteur standard SIRIS 8) :

- demande de connexion
- demande de déconnexion
- début de processeur - fin de processeur
- entrée d'information par décision d'un processeur
- entrée d'information par décision d'un usager

Articles à introduire dans le fichier :

- articles connexion :
  type d'article : 0 (4 octets)
  indicatif de l'usager (4 octets)
  date (20 octets)
- articles déconnexion
  type d'article : 1 (4 octets)
  indicatif de l'usager (4 octets)
  date (20 octets)
- articles début processeur
  type d'article : 2
  indicatif de l'usager (4 octets)
  date (20 octets)
  nom du processeur (4 octets)
- articles fin processeur
  type d'article : 3 (4 octets)
  indicatif usager (4 octets)
  date (20 octets)
  nom du processeur (4 octets)
- article processeur
  type d'article : 4 (4 octets)
  indicatif usager (4 octets)
  date (20 octets)
  nom du processeur (4 octets)

  information fournie (8 octets)
- article usager
  type d'article : 5 (4 octets)
  indicatif usager (4 octets)
  date (20 octets)
  information fournie (8 octets)

Principe de l'implémentation

Les ajouts suivants sont utilisés :

- une séquence au lancement permet d'ouvrir le fichier
- une séquence à la connexion et une séquence à la déconnexion permettent l'entrée des articles correspondants. L'indicatif de l'usager est l'extrait de la table TCT courante (désignée par le registre 1)
- trois séquences sur appels système permettent l'entrée des articles début et fin de processeur, et l'article processeur. L'indicatif de l'usager est extrait de la table TCT courante, qui fournit aussi un pointeur vers l'entrée correspondant au processeur actif dans la table TDP.
  L'appel correspondant à l'article début processeur devra être exécuté une seule fois par le processeur dès son lancement, celui correspondant à l'article fin processeur devra être exécuté une seule fois juste avant la fin (:EXIT). L'emploi du troisième appel est libre ; l'information associée est passée en paramètre.
- une commande immédiate permet l'entrée de l'article usager. L'indicatif de l'usager est extrait de la table TCT courante. L'information associée est fournie par lecture au terminal (appel au niveau TRTGSYS) dans le buffer associé à la tâche (son adresse est fournie par la table TCT) (rien ne sera mis dans le fichier en cas de mauvaise lecture).

Implémentation

```
        SYSTEM      SIG7
        SYSTEM      SIRIS7
        DEF         U≠INIT
        DEF         U≠CONN
        DEF         U≠DECON
```

```
          DEF         U≠DEPRO
          DEF         U≠FIPRO
          DEF         U≠PROC
          DEF         U≠USAG
          REF         RETRET
          REF         RETLI
          REF         RETLO
          REF         BRCHADR
          REF         TRTFCAL
          REF         TRTPROC
          REF         TRTGSYS
          REF         TPRONOM
DCB       M:DCB       (OPL,'CPT'),(REL,40),(FRM,V)
BU        RES,1       40
LNG       RES         1
M         TEXT        '****INF?'
*
*EXIT AU LANCEMENT
*
U≠INIT    M:OPEN      DCB,0
          B           RETRET
*
*RECHERCHE DE L'INDICATIF USAGER ET DE L'HEURE
*
USERD     EQU         15
FIXE      LW,3        USERD,1
          STW,3       BU+1
          M:TIME      BU+2
          B           *15

*
*RECHERCHE DU PROCESSEUR
*
PPROD     EQU         8
PROC      LW,3        PPROD,1
          LW,3        TPRONOM,3
          STW,3       BU+7
          B           *15
*
*ECRITURE DE L'ARTICLE
*
PUT       STW,7       LNG
          M:PUT       DCB,(REC,BA(BU)),(ARS,*LNG)
          B           *15
```



```
*
*EXIT A LA CONNEXION
*
U≠CONN    LI,3        0
          STW,3       BU
          BAL,15      FIXE
          LI,7        28
          BAL,15      PUT
          B           RETLI
*
*EXIT A LA DECONNEXION
*
U≠DECONN  LI,3        1
          STW,3       BU
          BAL,15      FIXE
          LI,7        28
          BAL,15      PUT
          B           RETLO

*
*APPEL DEBUT PROCESSEUR
*
U≠DEPRO   LI,3        2
          STW,3       BU
          BAL,15      FIXE
          BAL,15      PROC
          LI,7        32
          BAL,15      PUT
          MTW,1       BRCHADR
          B           TRTFCAL

*
*APPEL FIN PROCESSEUR
*
U≠FIPRO   LI,3        3
          STW,3       BU
          BAL,15      FIXE
          BAL,15      PROC
          LI,7        32
          BAL,15      PUT
          MTW,1       BRCHADR
          B           TRTFCAL

*
*APPEL PROCESSEUR
*
U≠PROC    LI,3        4
          STW,3       BU
```

```
          BAL,15        FIXE
          BAL,15        PROC
          MTW,1         BRCHADR
          LW,4          *BRCHADR
          CI,4          0
          BGE           S+2
          LW,4          *4
          LD,2          *4
          STD,2         BU+8
          LI,7          40
          BAL,15        PUT
          MTW,1         BRCHADR
          B             TRTFCAL
*
*COMMANDE IMMEDIATE
*
ICONORM   EQU           6
ABUFD     EQU           11
U≠USAG    LW,4          ABUFD,1
          LI,5          BA(M)
          LI,3          8
          BAL,15        TRTGSYS
          CI,0          ICONORM
          BNE           TRTPROC
          LW,4          ABUFD,1
          SLS,4         -2
          LW,2          0,4
          LW,3          0,4
          STW,2         BU+7
          STW,3         BU+8
          LI,3          5
          STW,3         BU
          BAL,15        FIXE
          LI,7          36
          BAL,15        PUT
          B             TRTPROC
*
*
*
          END
```


Génération du système modifié

Les directives suivantes doivent être ajoutées à la génération standard :

```
          REF           U≠INIT
          REF           U≠CONN
          REF           U≠DECON
          REF           U≠DEPRO
          REF           U≠FIPRO
          REF           U≠PROC
          REF           U≠USAG
          :EXIT         (ADR,U≠INIT)
          :EXLI         (ADR,U≠CONN)
          :EXLO         (ADR,U≠DECON)
          :CAL          (ADR,U≠DEPRO)
          :CAL          (ADR,U≠FIPRO)
          :CAL          (ADR,U≠PROC
          :COM          (NOM,'SSSS'),(ADR,U≠USAG)
```

L'écriture de l'article début processeur est alors obtenue par l'emploi de l'instruction :

```
          CAL2,0        12
```

L'écriture de l'article fin processeur est alors obtenue par l'emploi de l'instruction :

```
          CAL2,0        13
```

L'écriture des articles processeur est alors obtenue par l'emploi de l'instruction :

```
          CAL2,0        14
          GEN,1,31      i,a
```

où, si i=0, a est l'adresse du double-mot contenant l'information à transmettre

si i=1, a est l'adresse d'un mot contenant l'adresse du double-mot contenant l'information à transmettre

L'écriture des articles usager est alors obtenue par demande d'exécution de la commande SSSS. L'information transmise sera les 8 premiers caractères frappés en réponse à la question :

```
          ****INF ?
```

A N N E X E E

Mise en oeuvre

Mise en oeuvre de la génération de système

Deux fichiers sont disponibles pour la génération :

- le fichier partitionné contenant les "systèmes" METASYMBOL utilisés par la génération du système et des processeurs.
- le fichier séquentiel contenant le module objet du texte du système.

La génération des processeurs se met en oeuvre comme suit :

```
!JOB SYSDPROC,...
!LIMIT (TIME,...),(CORE,50),(SPDISC,...),(PAGES,...)
!ASSIGN PAR,MTN,FIL,(STS,OLD),(NAM,fichier "systèmes")
!METASYM SI,LO,GO,OL(PAR)
                :
                source 1er processeur
                :
!LINK (MAP)
!ASSIGN SWAP,FIL,(NAM,fichier des processeurs),(SIZ,...,...)
!ASSIGN GO,FRE
!RUN
!ASSIGN LM,FRE
!METASYM SI,LO,GO,OL(PAR)
                :
                source 2° processeur
                :
!LINK (MAP)
!ASSIGN GO,FRE
!ASSIGN SWAP,FIL,(STS,MOD),(NAM,fichier des processeurs)
!RUN

etc...
```

La génération du système se met en oeuvre comme suit :

```
!JOB SYSDSYST,...
!LIMIT (TIME,...),(CORE,50),(SPDISC,...),(PAGES,...)
!ASSIGN PAR, FIL,(STS,OLD),(NAM,fichier"systèmes")
!METASYM SI,LO,GO,OL(PAR)
          :
          :  procédures de description
          :
!ASSIGN E1,FIL,(STS,OLD),(NAM,module objet système)
!   ...assignation des ajouts éventuels (modules objets)
!ASSIGN LM,FIL,(NAM,fichier module de chargement système)
!LINK
:OPTION (MAP)
:TREE GO - E1 ajouts éventuels
```

Le lancement s'effectue comme suit :

```
!JOB,P SYSD,...
!LIMIT (TIME,500),(CORE,...),(SPDISC,...),(PAGES,...)
!ASSIGN SWAP,MTN,FIL,(SIZ,...,...)
!ASSIGN PROC,FIL,(STS,OLD),(NAM,fichier des processeurs)
!ASSIGN CNS,DEV,(LIN,...),(LLK,CNS1)
!ASSIGN CNS1,assignation de la liste des terminaux
!   ...assignations utilisées par les ajouts
!SWITCH (S,1)
!RUN (LMN,module de chargement système)
```

N.B. : - Tous les paramètres laissés blancs sont à évaluer en fonction de la génération et de l'utilisation.

- L'extension d'accès aux fichiers utilise l'assignation suivante :
  !ASSIGN REF,MTN,FIL,(STS,OLD),(NAM,:FDY),(UNT,support)

A N N E X E F

interface VCAM

Modifications dûes à l'utilisation de la méthode d'accès VCAM

Modification des principes

- Les terminaux réels étant remplacés par des correspondants virtuels, la notion de longueur de ligne n'a plus de signification : les périphériques sont gérés par les "gérants d'appareils" sur les sites, qui assurent le fractionnement des messages en cas de nécessité.
  Mais la méthode d'accès VCAM utilise un paramètre limitatif, la longueur maximum des messages échangés. Par : longueur de ligne, il faut entendre : longueur de message maximale.
- Les messages échangés ne sont plus des chaînes de caractères, mais des textes formatés conformément au protocole : appareil virtuel de VCAM. Les messages sont formés d'un ou plusieurs éléments de l'une des formes suivantes (L est la longueur totale de l'élément, en octets) :

L	X'01'	EBCDIC

Le texte est imprimé à partir de la position courante

L	X'02'	EBCDIC

Le texte est imprimé après un retour à la ligne

L	X'03'	EBCDIC

Le texte est imprimé après un retour à la page

L	X'04'	n	EBCDIC

Le texte est imprimé après impression de n blancs

L	X'CF'	n	X	EBCDIC

Le texte est imprimé après impression de n caractères de code X.

Les messages envoyés et reçus par les niveaux TRTPSYS, TRTPUSE, TRTGSYS, TRGUSE doivent respecter ce format.

- Le formatage des messages (retour à la ligne ou non en fin de message) n'est plus traité par la méthode d'accès, mais par le contenu des messages eux-mêmes.
  Les fonctions SAFRAN de formatage sont traitées à l'occasion des appels-système d'écriture en fonction d'une variable logique contenue dans la table TCT (FRMD).
- Le deuxième mot de l'indicatif usager obtenu par appel-système contient l'indicatif dans l'octet de faible poids et non plus de fort poids.

Modification des tables

- La table TRG a été modifiée :
  La variable NEVWA a été ajoutée après la variable NBCNS.
  Elle contient le nombre de tâches gérables plus une.
- La table TCT a été modifiée :
  La variable CONSD contient dans l'octet de faible poids, l'index dans la table des connexions VCAM, de la connexion correspondant à la tâche.
  La variable FRMD utilisée pour le formatage des messages a été insérée après AUX.
- Les événements ne sont plus gérés par la méthode d'accès, mais par la tâche. A chaque tâche sont associés deux événements contenus dans deux variables :
  - un pour les opérations bloquantes
  - un pour la surveillance parallèle
- La table des connexions de VCAM a été créée :
  Elle permet d'associer à chaque connexion une tâche du moniteur.
  Elle est utilisée par l'interface VCAM.
  Chaque TCT contient un pointeur vers une entrée de cette table (CONSD).
  Chaque entrée contient les variables suivantes :
  VCCON : variable logique indiquant si une connexion est en cours pour la tâche correspondante.
  VCREF : identification VCAM de la connexion. L'identification usager de la connexion est l'index dans la table : ceci permet de raccorder les séquences d'exception à la bonne tâche sans recherche en table.
  VCECB : adresse de l'événement bloquant.
  VCECX : adresse de l'événement de surveillance parallèle.
  VCCODX : niveau du dernier attention émis (1 ou 2).
  VCFATT : variable logique indiquant si l'arrivée d'un attention doit poster l'événement bloquant (désistement sur sémaphore).
  VCCODE : code d'achèvement de la dernière opération VCAM.
  VCBUF, VCREL, VCFLAC : variables de travail pour l'opération de lecture.

1) à toute demande de connexion, une entrée vide est choisie dans cette table (test de VCCON). La tâche correspondante est activée par postage de son événement bloquant (VCECB).
2) les demandes du moniteur SAFRAN à l'interface VCAM précisent toujours l'entrée dans la table correspondante (extraite de la table TCT : CONSD).

BIBLIOGRAPHIE

I : M. GRIFFITHS, M. PECCOUD, M. PELTIER

Incremental Interactive Compilation

Proc. IFIP August 1968

II : M. C. DENDIEN

Simulation de machines pour l'enseignement de la programmation

These. NANCY. Juin 1971

III : Moniteur TS SIRIS 8 CIIEB

Manuel d'utilisation et d'operations

IV : STRATEGE SIRIS7/8 CIIHB

Manuel d'utilisation et d'operations

V : Gestion des Transmissions SGT SIRIS 8 CIIEB

Manuel d'utilisation

VI : R. M. WATSON

Time Sharing System Design Concepts

Mc Graw Hill Book Company. Computer Science Serie 1970

VII : J. ARSAC

Systemes de Conduite des Ordinateurs

Dunod 1970

VIII : CROCUS

Systemes d'exploitation des Ordinateurs

Dunod 1975

IX : E. W. DIJKSTRA
Programming Languages
Genuys. Academic Press 1967

X : Procedures Systeme SIRIS 8 CIIHB
Manuel d'utilisation

XI : METASYMBOL SIRIS7/8 CIIHB
Manuel d'utilisation et d'operations

XII : Systeme de gestion de fichiers SGF SIRIS 8 CIIHB
Manuel d'utilisation et d'operations

XIII : La Methode d'acces VCAM SIRIS 8 CIIHB
Manuel d'utilisation

NOM DE L'ETUDIANT : *DUBUIT Michel*

NATURE DE LA THESE : *THESE DE DOCTEUR INGENIEUR en INFORMATIQUE*

VU, APPROUVE

et PERMIS D'IMPRIMER

NANCY LE *20 juin 1978* 4671

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

M. BOULANGE