MASTER EN SCIENCES INFORMATIQUES

Mesures de performances : construction d'un outil de mesures et modélisation du DEC-20

Rabeux, Christine; Pluquet, Gaëtan

Award date:
1984

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

INSTITUT D'INFORMATIQUE

Mesures de performances : Construction d'un outil de mesures et modélisation du DEC-20

Christine RABEUX

Gaëtan PLUQUET

Promoteur : Mme Noirhomme - Fraiture

Mémoire présenté en vue
de l'obtention du grade de
LICENCIE ET MAITRISE EN INFORMATIQUE

ANNEE ACADEMIQUE 1983 - 1984

Mémoire de Christine RABEUX
Gaëtan HINQUET

"Mesures de performances :
Construction d'un outil de mesures et
modélisation du DEC-20."

ANNEE 1983-1984

Nous nous excusons auprès des lecteurs de ce mémoire pour les différentes erreurs de dactylographie qu'ils pourraient rencontrer. Certains problèmes ne nous ont pas permis de relire assez en profondeur ce volume avant sa duplication; aussi, en vue de faciliter la compréhension du lecteur, avons nous dressé une liste des erreurs les plus importantes. Dans celle-ci, nous reprenons chaque fois une partie de la phrase où l'erreur se situe et soulignons le ou les mots erronés tels qu'ils doivent être lus.

p. 6. .... et de mémoire virtuelle avec pagination.

p. 10. (I.1.4.) - Courant : le processus occupe le processeur....

p. 18. (2° §) .... le faire passer dans l'état Prêt-Actif. Un processus en attente est donc un processus dans l'état Bloqué-Actif, Bloqué-Inactif ou Prêt-Inactif .....

p. 19. (3.) crédit + reste du "quantum" > 3000 ms .....

p. 24. (1° §) .... décrire la manière dont un processus ....
(dern.§) .... : le nombre maximum de processus lui appartenant, donc ....

p. 26. ((1)) .... disponible pour les processus utilisateurs ....
.... et volontairement du balance set lorsqu'il effectue ....

(1)) (Cette méthode est connue dans la ....
.... "Round Robin Method").

p. 28. (dern.§) .... scheduler qui décide quel processus à exécuter.

p. 29. (I.5.) ...., imprimantes, ... qui elles sont longues.

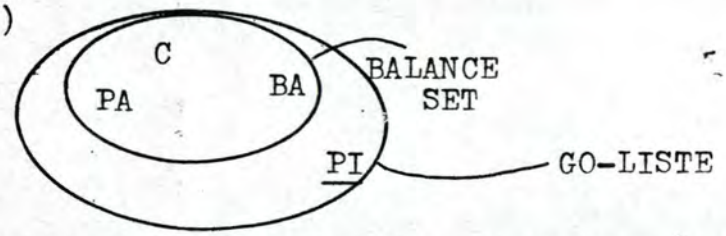
p. 30. (I.5.2.) .... préférence aux demandes de lecture. En favorisant la lecture, on tend à réduire le temps d'attente des processus dû au swapping car un processus attend généralement pour une lecture tandis que les écritures sont plutôt générées par la "Local Garbage Collection".

p. 31. .... supérieur pour lequel il existe une demande de lecture.
.... supérieur pour lequel il existe une demande d'écriture.
.... inférieur pour lequel il existe une demande de lecture.
.... inférieur pour lequel il existe une demande d'écriture.

p. 33. (I.6.1.) .... des valeurs s'étendant de 1 à 20, ...

p. 34. (2° §) .... et non chaque processus pris ....

p. 38. (fig.I.5.)



- Les processus Bloqués-Inactifs ne se trouvent pas dans la GO-LISTE, ...

p. 39. (fig.I.6.) La transition PA ⟵ BA doit être orientée dans ce sens.
La transition BA ⟶ BI ne doit pas exister dans cette figure.
Création d'un processus.

p. 49. (1° §) .... centrale (file devant Δ)....

p. 51. ((3)Rem.) .... selon les niveaux de priorité des processus,...

p. 59. (1° §) .... du bras de lecture) (cfr.I.5.2.).

p. 77. (2° §) .... cumuler sur "i" les "λi" et "$\bar{\mu}$i".

p. 87. (3° §) .... dans les listes d'attente pour entrées/sorties sur terminal .....

p. 92. .... plus précis vu que la taille de l'échantillon. ...

p. 117. (IV.2.5.) 4) Quand le sous-modèle contient plusieurs chaînes fermées.

p. 124. (2° §) .... théorie de la décomposabilité presque complète (cfr.IV.2.6.): ....
.... (.... l'autre la synchronisation des disques et canaux).

p. 129. (2° §) .... décompositions précédentes, le troisième niveau d'agrégation respecte la théorie de la décomposabilité presque complète, par contre, le deuxième ne la respecte pas.

p. 156. [VER] René Verhaegen

Nous tenons tout d'abord à exprimer notre reconnaissance à Madame NOIRHOMME-FRAITURE qui a accepté de diriger ce mémoire et qui nous a permis par ses conseils et critiques, de mener ce travail à son terme.

Nous remercions également Messieurs Michel DAWIRS, Jorge BARRETO et Michel DEBAR pour l'aide qu'ils nous ont prodiguée tout au long de ce travail.

Nous tenons aussi à remercier Monsieur Christian ADANS de "Digital Equipement Corporation".

Nous exprimons notre gratitude à Messieurs Bruno DURASSE, Gabriel LESTRATE et à tout le personnel du Centre de Calcul de l'Institut d'Informatique pour leur patience et leur disponibilité lors de la réalisation de ce travail.

Enfin, notre reconnaissance va à toute personne qui de près ou de loin a participé à la mise en page et l'édition de ce mémoire.

TABLE DES MATIERES

INTRODUCTION

L'analyse et l'évaluation des performances des systèmes acquièrent une place de plus en plus importante dans le domaine de l'informatique. De plus en plus on exige des systèmes informatiques qu'ils soient très puissants, très flexibles, très performants. On exige que les services rendus à l'utilisateur soient les meilleurs possibles pour sa productivité. A cause de la dimension des investissements que requiert l'installation d'un système informatique, on accorde une attention considérable à l'efficacité de celui-ci. Le coût élevé des conceptions "hardware" et des développements "software" crée une pression pour évaluer les systèmes proposés ou les changements projetés sur un système déjà existant avant même de commencer l'implémentation en elle-même.

Il existe différentes situations dans lesquelles les responsables de systèmes peuvent souhaiter une étude de performances.

- Améliorer les performances d'un système déjà existant ("performance tuning").
- Maintenir les performances d'un système d'exploitation dans des limites raisonnables déterminées ("performance control").
- Concevoir et implémenter un nouveau système et pour cela étudier les conditions d'exploitation et les performances des systèmes proposés selon certains critères (bench marks).

Il va de soi que les moyens à envisager sont très différents selon le type de situation dans lequel on se trouve.

La performance d'un système peut être caractérisée par son efficacité et son efficience. L'efficacité d'un système est décrite en termes de capacité à satisfaire la demande des utilisateurs pris individuellement. L'efficacité d'un système peut être estimée à partir d'observations faites sur sa partie externe : c'est le système tel qu'il est vu par l'utilisateur. (Elles sont appelées "mesures de performances orientées utilisateur"). Par contre l'efficience d'un système est mesurée dans sa partie interne. Ces mesures aident à identifier les problèmes qui diminuent les performances du système. (elles sont appelées "mesures de performances orientées système").

Estimer l'efficacité d'un système ne peut produire que des constatations sur le comportement de celui-ci, tandis que estimer l'efficience d'un système peut établir les raisons de son comportement.

Par exemple, le temps de réponse d'un système interactif est une mesure orientée utilisateur. Le temps de réponse représente le temps écoulé entre le moment où l'utilisateur envoit une demande vers le système et le moment où il reçoit la réponse à sa demande. Tandis que le pourcentage d'utilisation d'une ressource est une mesure orientée système.

Notre travail dans ce mémoire, est d'étudier les performances du DECSYSTEM-20 des Facultés universitaires de Namur. Il s'agit donc d'étudier un système soumis à une charge assez particulière et très irrégulière : elle est composée d'étudiants, de chercheurs, d'assistants, de professeurs. Toute analyse d'un système est seulement l'analyse d'un certain modèle du système. Un modèle est une abstraction du réel contenant uniquement les mécanismes importants et permettant à l'analyste de synthétiser le système et de se concentrer sur le principal. A travers la modélisation, les détails non nécessaires sont supprimés, les mécanismes du système sont simplifiés.

Notre travail consiste donc à proposer un modèle du DEC-20 et à l'implémenter. Ce modèle devrait pouvoir être utilisé ultérieurement pour tester les améliorations possibles du système. Nous nous trouvons donc dans la première des situations que nous venons de décrire ("performance tuning"), et devons mesurer l'efficience du système.

La mise au point de ce travail est assez complexe et il ne nous a pas toujours été possible d'approfondir tous les détails de sa réalisation avec les mêmes soins que ceux susceptibles d'être attribués par une équipe de chercheurs.

Néanmoins, en limitant d'avance notre travail, un tel but nous a semblé réalisable.

Notre étude s'est déroulée en plusieurs étapes. Tout d'abord il nous fallait acquérir de bonnes connaissances du système informatique DECSYSTEM-20. Nous pouvons considérer un ordinateur comme étant une entité constituée de multiples composants. Des travaux demandent des services à l'ordinateur et un service doit être représenté par au moins
- des demandes "CPU"
- des demandes d'espace mémoire
- des demandes d'Entrées-Sorties.

Le chapitre I présente une description du système réunissant tous ces composants. Il contient les concepts de base nécessaires pour comprendre le système et effectuer les mesures.

Le chapitre II est consacré à la modélisation du système et à la justification du modèle par rapport au système réel. Dans ce chapitre, nous déterminerons également les paramètres nécessaires pour réaliser ce modèle.

Des informations exactes sur le comportement actuel du système ne peuvent être obtenues qu'en effectuant des mesures sur le système "actif". Sans ces mesures réelles, aucune hypothèse de performance ne peut être validée. Il est donc indispensable de mesurer les paramètres à fournir au modèle sur le système "actif". Dans le chapitre III, nous exposons la manière dont nous avons procédé pour effectuer ces mesures avec les outils que nous avions à notre disposition. C'est la construction de ce mesureur qui a pris une grande partie du temps consacré au mémoire.

Enfin, à partir des performances du système réel, nous pouvons dès lors résoudre notre modèle valablement. Ce modèle pourra par la suite être utilisé pour envisager une amélioration du système.

Le chapitre IV montre comment le modèle a pu être implémenté et résolu.

CHAPITRE I

ETUDE DU SYSTEME

I.0. INTRODUCTION

Le système que nous étudierons ici est le système d'exploitation TOPS-20 tel qu'il est implémenté au centre de calcul des Facultés Universitaires Notre Dame de la Paix à Namur : c'est-à-dire TOPS-20 dans sa cinquième version. Dans ce centre, ce système s'exécute sur un ordinateur DECSYSTEM 20 (DIGITAL) qui est constitué d'un processeur KL10 couplé avec un ordinateur frontal PDP-11 chargé de gérer la console opérateur et les terminaux. TOPS-20 est un système d'exploitation classique qui permet d'exécuter des tâches en temps réel, en mode interactif et en lots (batchs), et ce, en utilisant les techniques bien connues de temps partagé et de mémoire mutuelle avec pagination.

Dans un premier temps, nous décrirons quelques concepts de bases de ce système (I.1.) qui nous permettront ensuite de décrire les éléments nécessaires pour la compréhension de la modélisation et du mesureur :

- la manière dont les processus sont gérés (I.2.),
- la manière dont la mémoire centrale est gérée (I.3.),
- la manière dont le CPU est alloué aux utilisateurs (I.4.),
- la manière dont les entrées/sorties sur disques et bandes sont prises en charge (I.5.),
- les options de contrôle de ce système (I.6.).

A partir de la description de ces éléments du système, nous pourrons donner une signification plus pratique à certains concepts de base (I.7.).

I.1. CONCEPTS DE BASE [DIG5, VER]

Pour comprendre le système qui nous intéresse, commençons par décrire quelques concepts généraux.

I.1.1. Comment le système répond-t-il aux besoins de l'utilisateur ?

On peut considérer l'ordinateur comme un ensemble de ressources que le système essaye d'allouer aux utilisateurs. Cette allocation doit être faite de manière telle que l'ordinateur soit utilisé de la manière la plus efficace possible (partage équitable entre les processus, pourcentage élevé d'utilisation du CPU, ...) et que le temps de réponse aux demandes de l'utilisateur soit acceptable.

Les ressources à prendre en considération sont :

- le processeur central (CPU)
- la mémoire centrale
- les unités d'E/S (bandes, imprimantes, terminaux ...)
- les fichiers.

Certaines de ces ressources peuvent être partagées par plusieurs utilisateurs : mémoire, fichiers.

Tandis que d'autres ne peuvent être assignées qu'à un seul utilisateur à la fois : CPU, imprimante, ...

Outre l'allocation des ressources, le système d'exploitation est chargé d'effectuer d'autres services pour les utilisateurs :

- la gestion des tâches des utilisateurs (ordonnancement des processus)
- le décodage de commandes des utilisateurs
- la gestion des erreurs.

Pour assurer l'intégrité du système, certaines fonctions ne peuvent être exécutées directement par les utilisateurs : il faut faire appel au système qui les gérera lui-même. Il s'agit de fonctions qui visent à

- obtenir des informations sur les opérations du système
- obtenir des informations sur les autres utilisateurs
- effectuer des entrées/sorties.

Ces services sont effectués par le système à la demande de l'utilisateur (opérations d'E/S, obtention d'information), ou sont activés régulièrement par une horloge interne (décodage des commandes, ordonnancement des tâches, ...).

I.1.2. La base de données du moniteur

Pour accomplir toutes ces fonctions, le système doit maintenir une grande quantité d'informations dans une base de données. Il conserve des tables pour chaque "job" dans son ensemble (Job Tables), des tables pour chaque processus (Fork Tables), des tables d'informations pour chaque unité, chaque canal, et beaucoup d'autres tables pour le système qui découleront des concepts expliqués ci-après. Les principales tables de cette base de données sont décrites dans l'annexe A.

I.1.3. "Jobs" et Processus

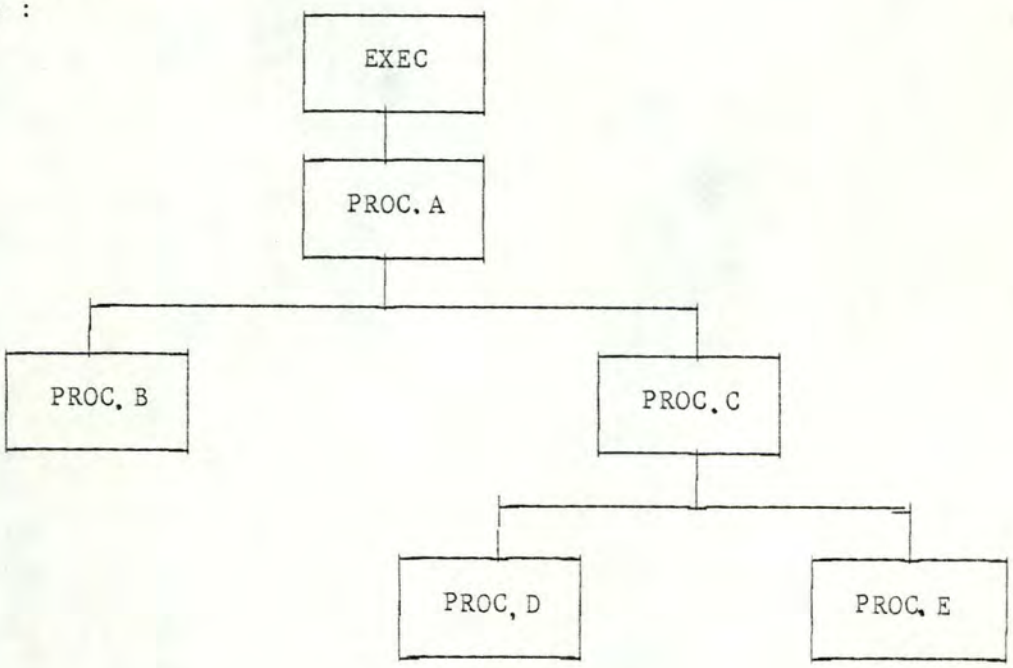
Quand un utilisateur veut travailler, il introduit un CTRL/C à un terminal. Le moniteur crée alors un "Job" et associe le terminal à ce job.

Quand l'utilisateur introduit LOGIN, le "job" et le terminal sont associés avec l'utilisateur

- à un "job" correspond un utilisateur,
- à un "job" correspond un terminal,
- à un terminal correspnd un job,
- à un terminal correspond un utilisateur,
- mais à un utilisateur peuvent correspondre plusieurs "jobs" car un utilisateur peut travailler sur plusieurs terminaux.

Accomplir le "job" d'un utilisateur peut impliquer l'exécution de plusieurs tâches. Ces différentes tâches peuvent être exécutées simultanément, séquentiellement ou alternativement. Elles sont effectuées par des processus. Un processus est un programme qui entre en compétition avec d'autres pour obtenir différentes ressources. Le moniteur ne s'occupe pas des besoins du "job" dans son ensemble, il essaye de satisfaire individuellement les besoins des différents processus et on suppose qu'il satisfait ainsi les besoins du "job".

Les processus d'un "job" sont organisés selon une structure d'arbre : chaque processus peut en créer d'autres.
exemple :



Le processus de niveau le plus élevé est l'interpréteur de commandes : EXEC. Quand l'utilisateur fait exécuter un programme, ce programme devient un processus descendant de EXEC. Ce processus peut lui-même en créer un ou plusieurs autres. Chaque processus peut contrôler plusieurs fils, mais un processus ne peut avoir qu'un père. Un père a tout pouvoir sur ses fils qu'il peut démarrer, arrêter, et suspendre à volonté. Tandis qu'un fils ne peut manipuler son père qu'avec l'autorisation préalable de celui-ci.

I.1.4. Les états d'un processus

Un processus peut se trouver dans différents états en fonction des décisions du moniteur et du cours de son exécution. Ces états sont le résultat de la combinaison des trois situations suivantes :

- Bloqué/Prêt : Le processus attend ou n'attend pas la réalisation d'un événement autre que l'attribution de la ressource mémoire centrale. Il est en général bloqué en attente de la réalisation d'une entrée/sortie.
- Actif/Inactif : Le processus a le droit d'occuper une partie de la mémoire centrale ou n'a pas ce droit. Il possède ou non la ressource mémoire centrale.
- Courant : Le processus occupe le processus, il possède la ressource CPU.

A ces 3 situations, on peut ajouter deux autres situations limites : le processus non encore créé et le processus détruit.

Des combinaisons des trois situations principales découlent les états possibles pour un processus.

BA : Le processus possède la ressource mémoire centrale, mais est bloqué pour une autre raison.

PA : Le processus possède toutes ses ressources et peut être sélectionné pour s'exécuter, il peut devenir courant.

BI : Le processus ne possède pas la ressource MC mais est également bloqué pour une autre raison.

PI : La seule chose qui manque au processus pour pouvoir être sélectionné pour s'exécuter est la ressource MC.

C : Le processus s'exécute, il occupe le CPU.

Nous ferons, dans la dernière partie de ce chapitre (I.7.) une synthèse de la signification de ces états par rapport au système décrit et examinerons les différentes transitions entre chaque état.

I.2. GESTION DES PROCESSUS [DIG2, DIG5]

Nous allons dans ce paragraphe examiner comment les processus sont gérés par le système :

Que se passe-t-il quand un processus est créé (I.2.1.) ?

Comment sont classés les processus (I.2.2.) ?

Comment leur est attribuée une priorité (I.2.3.) ?

Que se passe-t-il à la terminaison d'un processus (I.2.4.) ?

Ajoutons que c'est le module du système "scheduler" qui effectue la plupart de ces tâches.

I.2.1. Création de "jobs" et de processus

Quand on crée un "job" le système crée automatiquement des tables pour ce "job" et met à jour les tables existantes. Cette création implique la création d'un processus, qui peut lui-même en créer d'autres. Pour chaque processus, le système construit également des tables et des entrées dans les tables existantes. Sans ces tables, le processus ne peut rien faire, il n'existe pas.

Il existe, cependant un cas un peu plus particulier : les travaux en lots souvent appelés "batchs". Un "batch" est un "job" dont les entrées sont complètement comprises dans un fichier. Le "batch" peut être créé à partir d'un terminal ou bien être le résultat de la lecture de cartes perforées. La prise en charge de ces jobs respecte les étapes suivantes :

1. assigner un pseudo-terminal (PTY) au "batch job".
2. entrer un CTRL/C au "PTY".
3. effectuer le "login" de l'utilisateur.
4. lire le fichier de commandes.

A partir de là, le "batch job" est considéré comme un autre "job". Les "batch jobs" sont toujours "compute-bound" (par opposition à interactif) puisqu'il ne peut y avoir d'interaction avec l'utilisateur à son terminal.

I.2.2. Différentes listes de processus

Pour traiter tous ces processus, le système les classe dans différentes listes selon qu'ils sont en attente longue ou non et selon la raison de cette attente. Nous dirons qu'un processus est en attente longue s'il se trouve dans l'état Bloqué-Inactif (cfr. I.1.4.) Si un processus est bloqué pour un temps assez long, il sort de mémoire centrale (devient inactif).
Nous dirons qu'il n'est pas en attente longue s'il se trouve dans tout autre état.

1) la GO-LISTE (GLST)

Cette liste contient tous les processus qui ne sont pas en attente longue c'est-à-dire tous les processus qui sont dans un des états suivants :

- Prêt - Actif
- Bloqué - Actif
- Prêt - Inactif
- Courant.

La GLST doit être ordonnée selon les priorités des processus.
Nous verrons dans le point suivant (I.2.3.) comment sont attribuées et gérées ces priorités.

2) les listes d'attente

Les processus en attente longue sont classés dans plusieurs listes selon les différents types d'événements qui ont causés leur attente. Un processus en attente longue étant un processus dans l'état Bloqué - Inactif, cette classification nous éclaire sur la raison pour laquelle le processus se trouve dans cet état.
Ces listes sont au nombre de 7.
TTILST : liste des processus en attente pour une entrée sur terminal

TTOLST : liste des processus en attente pour une sortie sur terminal

FRZLST : liste des processus gelés. Un processus peut être gelé par son père, et le père peut modifier par exemple les accumulateurs de son fils avant de le dégeler pour qu'il continue son exécution.

CLKLST : liste des processus en attente pour une durée bien déterminée.

TRMLST : liste des processus en attente de la fin d'exécution de leurs fils.

WT2LST : liste des processus qui attendent pour un IPCF (Inter Process Communication Facility).

WTLST : liste des processus en attente pour d'autres raisons diverses.

Nous résumerons la classification des processus en listes par la figure I.1. montrant les transitions entre elles.

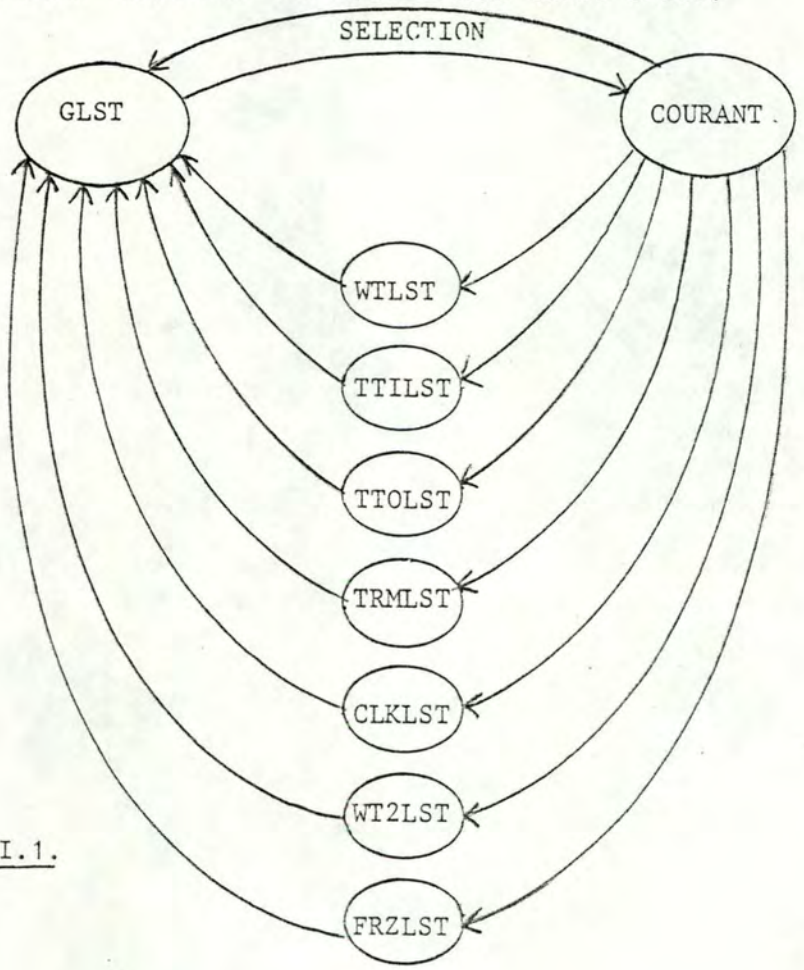


figure I.1.

Un processus appartient soit à la GO-LISTE, soit à une des listes d'attente. Seul un processus de la GO-LISTE peut être sélectionné pour s'exécuter. Quand l'exécution est terminée, il peut alors changer de liste.

I.2.3. Les files auxiliaires

Pour assigner des priorités aux processus de la GO-LISTE, le moniteur TOPS-20 utilise une structure de 6 files auxiliaires. Ces files sont numérotées de 0 à 5, la file 0 contenant les processus les plus prioritaires et la file 5 les processus les moins prioritaires. Ces files auxiliaires contiennent tous les processus du système qu'ils soient dans la GO-LISTE ou pas. Mais elles ne servent en réalité que pour attribuer des priorités à ceux de la GO-LISTE, les autres listes n'ayant pas besoin d'être ordonnées.

I.2.3.1. Le mouvement descendant

A chaque file est associée une durée maximale de maintien ("queue quantum). Un processus ne peut pas utiliser dans une même file plus de temps CPU que ce "quantum". Les files 1 à 4 se succèdent une à l'autre : quand un processus appartenant à une de ces files atteind son quantum, il est envoyé dans la file de priorité inférieure et son "quantum" est réinitialisé. La file 4 se succède à elle-même : un processus atteignant son "quantum" dans cette file est envoyé en fin de cette même file avec un "quantum" réinitialisé. Quand il a atteind la file 4, un processus, ne peut plus descendre de file.

Pour les processus des files 0 et 5, le mouvement est un peu différent. Ces files sont des files spéciales pour les processus de haute priorité et de basse priorité. Les processus entrant dans une de ces files n'en sortent pas.

Quand ils ont épuisé leur "quantum", ils rentrent dans la même file. Ces files se succèdent donc aussi à elles-mêmes.

Les différents "quantum" (temps de maintien) pour chaque file sont les suivants :

file 0 : 300 ms.
file 1 : 300 ms.
file 2 : 1.000 ms.
file 3 : 3.000 ms.
file 4 : 3.000 ms.
file 5 : 10.000 ms.

La figure I.2. schématise le mouvement descendant des processus. Lorsqu'un nouveau processus entre dans la GO-LISTE, il est placé dans la file 1 et sa position est réévaluée chaque fois qu'il s'exécute un certain temps et chaque fois qu'un processus entre dans la GO-LISTE.

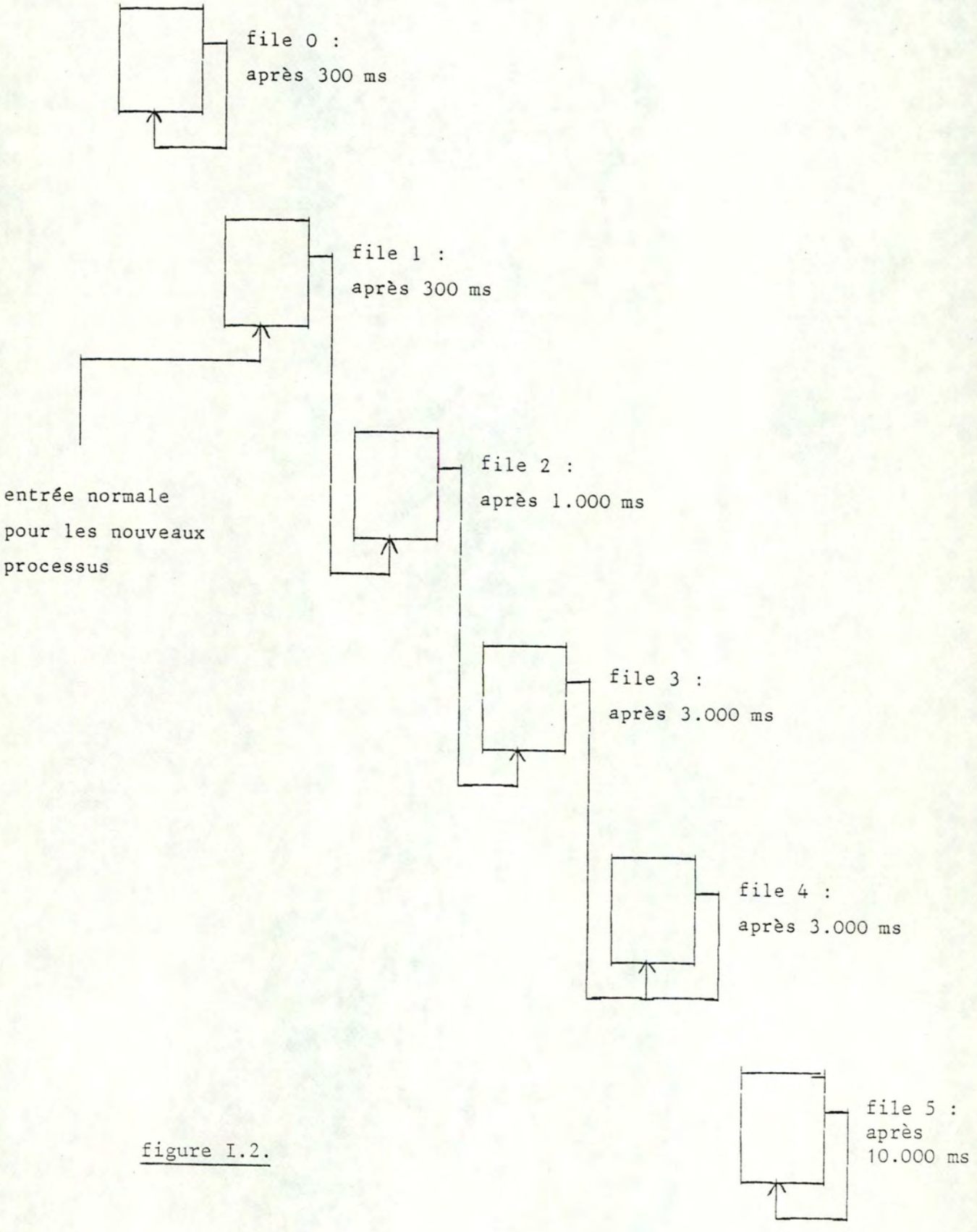

file 0 :
après 300 ms
file 1 :
après 300 ms
entrée normale
pour les nouveaux
processus
file 2 :
après 1.000 ms
file 3 :
après 3.000 ms
file 4 :
après 3.000 ms
file 5 :
après
10.000 ms

figure I.2.

I.2.3.2. Le mouvement ascendant

Le "scheduler" peut également procéder pour certains processus à des transitions vers des files auxiliaires de priorités supérieures. On considère en effet qu'une attente doit modifier la priorité du processus en lui étant favorable, donc augmenter sa priorité en faisant passer le processus vers une file de numéro inférieur.

Par attente nous entendons ici, l'attente pour un processus d'un événement ou de ressources qui pourraient le faire passer dans l'état Prêt. Un processus en attente est donc un processus dans l'état Bloqué-Actif ou Bloqué-Inactif (et pas uniquement attente longue, B - I au sens de I.2.2.).

Pour comprendre ces changements de files, nous allons introduire la notion de "n-minute-load-average".
Le "load average" est une estimation du nombre de processus dans la GLST. La longueur instantanée de cette liste, variant considérablement à chaque instant, elle ne peut être retenue comme estimation. Le "load average" est calculé régulièrement, après n minutes et dépend de deux valeurs :

- le "load average" précédent
- l'intégration de la charge (longueur de la GLST) pendant les n dernières minutes.

Ces "n-minute-load-average" sont calculés pour n égal à 1,5 et 15 minutes et permettent d'avoir une idée du temps que le programme prendra pour s'exécuter.

Pour savoir quel changement de file il faut faire, il faut calculer un "crédit d'attente" . Si l'attente est courte, aucun crédit n'est donné (crédit = 0). Tandis que si l'attente est longue et que le "1-min-load-average" est supérieur à 2, le crédit est inversément proportionnel à ce "load average". Si le "1-min-load-average" est inférieur à 2, le crédit est égal au temps d'attente.

Les **règles** de changement de files sont les suivantes et sont appliquées quand un processus de la GLST passe dans l'état Prêt - Actif.

1. Si le processus se trouvait dans une des files 0, 1 ou 2, il y reste : on considère qu'il a une priorité suffisamment grande.
2. Si le processus se trouvait dans la file 5, il y reste. La fi-5 contient des processus spéciaux qui doivent rester dans cette file.
3. Si le processus se trouvait dans une des files 3 ou 4 et était en attente pour E/S sur terminal, le changement s'effectue comme suit :

   Si la somme du crédit et du reste du "quantum" pour ce processus est supérieur au "quantum" de la file, alors le processus est envoyé dans la file de priorité supérieure.

   Donc si

   crédit + reste du "quantum" > 300 ms

   il est envoyé dans la file de priorité supérieure.
4. Si le processus était en attente pour une raison autre qu'une E/S sur terminal et qu'il se trouvait dans le file 3, il ne change pas de file.

   Si il se trouvait dans la file 4, et que son "crédit d'attente" est plus grand que 6.000 ms, il est envoyé dans la file 3.

La figure I.3. schématise le mouvement ascendant des processus.

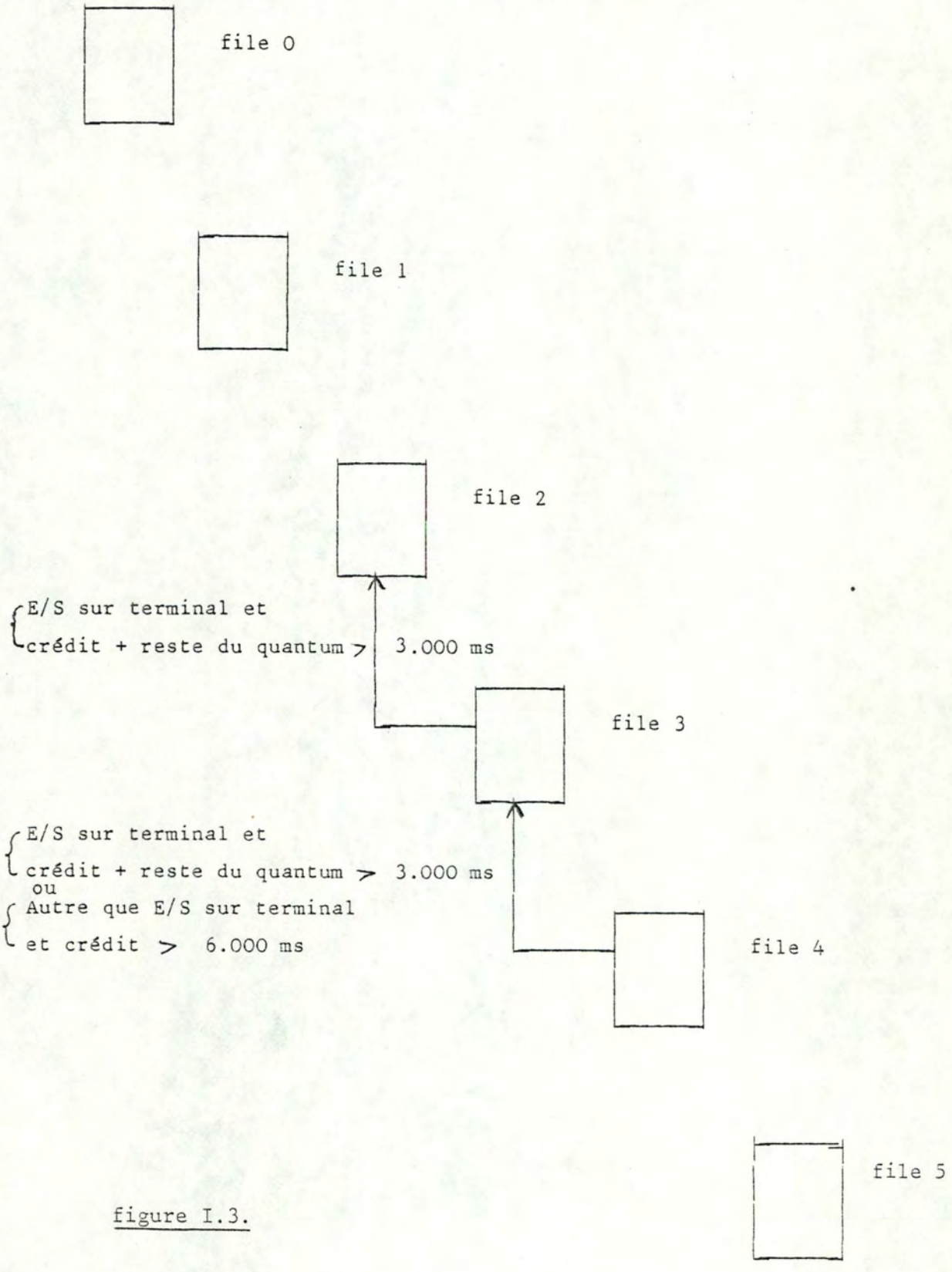

file 0
file 1
file 2
E/S sur terminal et
crédit + reste du quantum > 3.000 ms
file 3
E/S sur terminal et
crédit + reste du quantum > 3.000 ms
ou
Autre que E/S sur terminal
et crédit > 6.000 ms
file 4
file 5

figure I.3.

I.2.3.3. But d'une structure en files auxiliaires

Grâce aux files auxiliaires et aux mouvements à l'intérieur de celles-ci, on classe les processus en fonction de leur activité dans le système.

- la **file 0** est utilisée par les processus qui requièrent une attention particulière.
- les processus interactifs restent dans les **files 1 et 2** . Il y a 2 files interactives parce que les processus interactifs sont de deux types : ceux qui s'exécutent en temps très court avant de se bloquer et ceux qui s'exécutent en temps un peu plus long. Les "quantum" pour ces files reflètent cette différence. Les processus les plus interactifs se trouvent dans la file 1.
- la **file 3** contient les processus intermédiaires entre interactifs et "compute-bound".
- les processus les plus "compute-bound" convergent vers la **file 4** .
- la **file 5** contient des processus qui ne sont jamais interactifs.

Ces techniques permettent de privilégier les processus faisant réellement un travail interactif, d'éviter de trop longues attentes pour les processus les moins prioritaires et de tenir compte des délais d'attente pour obtenir une bonne rotation des processus dans la GO-LIST.

La figure I.4. résume les mouvements et le but des files auxiliaires.

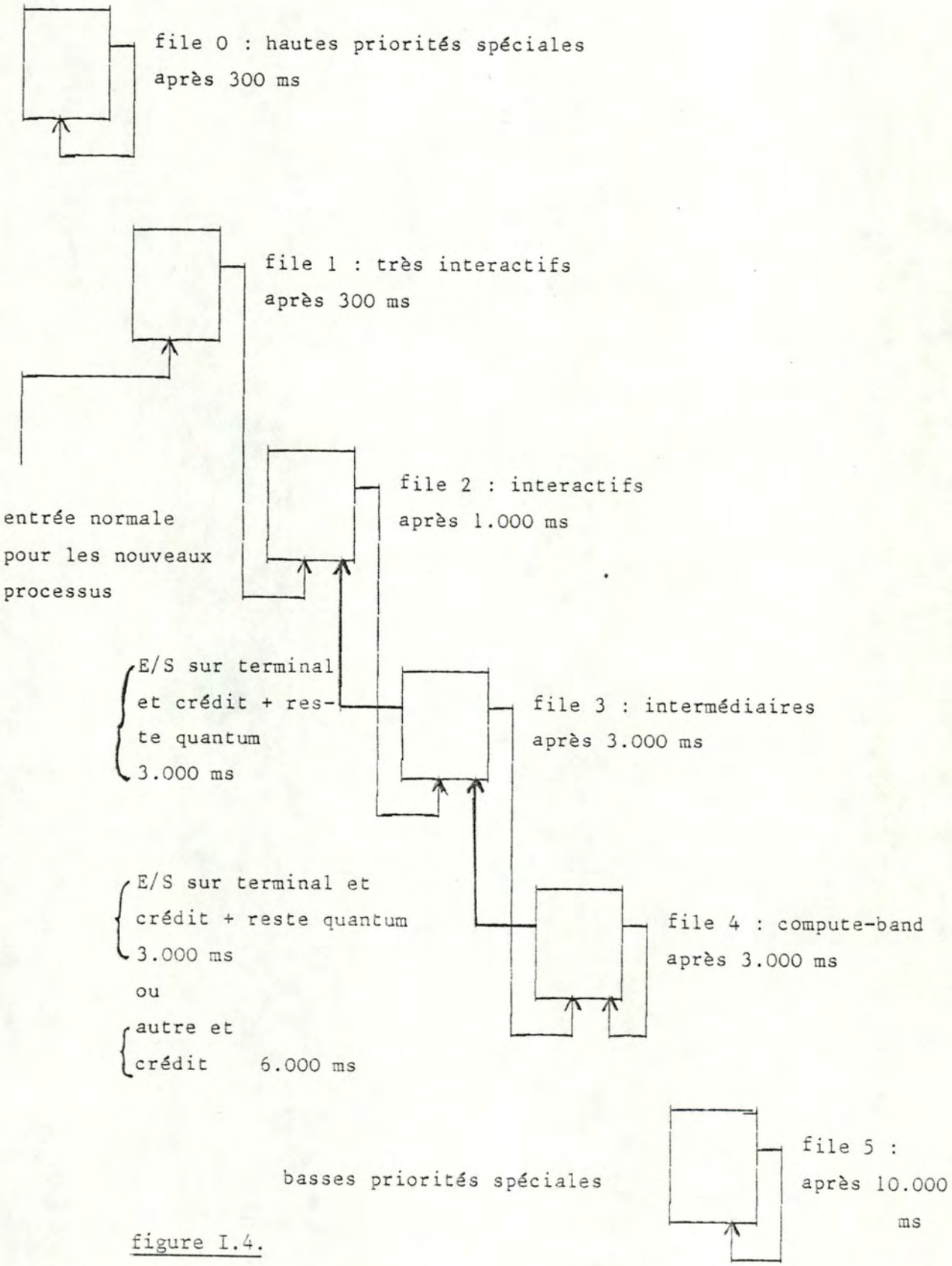

file 0 : hautes priorités spéciales
après 300 ms
file 1 : très interactifs
après 300 ms
entrée normale
pour les nouveaux
processus
file 2 : interactifs
après 1.000 ms
E/S sur terminal
et crédit + res-
te quantum
3.000 ms
file 3 : intermédiaires
après 3.000 ms
E/S sur terminal et
crédit + reste quantum
3.000 ms
ou
autre et
crédit 6.000 ms
file 4 : compute-band
après 3.000 ms
basses priorités spéciales
file 5 :
après 10.000
ms

figure I.4.

I.2.4. Terminaison

Il y a deux étapes pour traiter la terminaison. La première se produit quand le processus a terminé son exécution et la deuxième lors du "nettoyage" de la base de données du système.

1ère étape

Lors de la terminaison normale ou anormale du processus, le moniteur effectue les opérations suivantes :

1. M-A-J des temps d'exécution du processus sortant.
2. Réveil du processus supérieur s'il attendait la terminaison de ce processus.
3. Mise du processus sortant dans le liste d'attentes diverses (WTLST)
4. Sélection d'un nouveau processus.

2ème étape

Elimination totale du processus en effectuant les opérations suivantes :

1. Nettoyage des fichiers que le processus a utilisés.
2. Libération de la mémoire physique utilisée.
3. Libération des pages mémoires occupées par les tables du processus.
4. Mise à jour de la structure du "job".
5. Enlèvement des entrées dans les différentes tables du moniteur.

Ces 2 étapes de terminaison sont séparées pour permettre aux processus supérieurs de redémarer ou d'examiner le processus sortant.

I.3. GESTION DE LA MEMOIRE CENTRALE [DIG1, DIG2, DIG5]

Nous allons maintenant décrire la mémoire dont un processus s'octroie une part de mémoire, la manière donc dont il devient actif (I.3.2.) et la manière dont cette part de mémoire est gérée (I.3.1.).

Pour comprendre ces mécanismes, il nous faut d'abord énoncer deux notions importantes qui sont la base de ce paragraphe : le "working set" et le "balance set".

* le **"working set"** d'un processus est l'ensemble des pages dont il a besoin en mémoire centrale pour s'exécuter. En effet, grâce à la mémoire virtuelle, le processus n'a pas besoin de toutes ses pages en mémoire centrale. S'y trouvent uniquement celles qu'il utilise ou vient d'utiliser, les autres étant mémorisées de manière virtuelle. Le "working set" est donc l'ensemble des pages d'un processus les plus récemment référencées.
  A cette notion de "working set" s'en ajoutent deux autres :
  WSS = working Set Size = le nombre maximum de pages que le processus peut avoir en M.C.
  C'est une tentative de prévision du nombre de pages nécessaires pour que le processus tourne de manière satisfaisante (pas trop de défauts de pages).
  CWSS = curent working Set Size = le nombre de pages que le processus possède réellement en mémoire centrale à un moment donné.

* le **"balance set"** du système est l'ensemble des processus ayant leur working set en mémoire centrale. En d'autres mots, c'est l'ensemble des processus actifs. Tous ces processus sont repris dans une liste du système : BALSET qui est un sous-ensemble de la GO-LISTE.
  Toutefois, le système impose une restriction au BALSET : le nombre minimum de processus lui appartenant, donc le nombre maximum de processus actifs, est de 50.

I.3.1. La gestion du "working set"

La taille courante du "working set" (CWSS) est contrôlée par 2 routines : le "local garbage collection" et la "globale gabage collection". En effet, un processus émet des défauts de pages, ces pages sont lues en M.C. et augmentent la taille du "working set". Il est indispensable de contrôler cette taille pour qu'elle n'augmente pas démesurément.

* **La local garbage collection** est une action qui calibre la CWSS pour un processus. Elle garde en M.C. uniquement les pages les plus récemment référencées et réécrit les autres sur disque.

* **La global garbage collection** est exécutée pour être sûr que la mémoire ne soit pas gaspillée. Elle est exécutée globalement pour tous les processus. C'est une action qui collecte des pages soit, dont on a perdu la trace (le processus n'est plus en M.C.) ou soit qui étaient partagées et qui ne sont plus utilisées.

I.3.2. La gestion du "balance set"

Le "balance set" est mis à jour de deux manières différentes : soit dynamiquement, soit périodiquement, mais toujours par le module "scheduler" du système.

Le choix des procesus à faire entrer dans le "balance set" est fait **dynamiquement** lorsqu'un processus entre dans la GO-LISTE et lorsqu'un processus quitte le "balance set". En d'autres mots, on détermine si il faut faire entrer un processus en mémoire, et si oui, lequel, quand un processus passe de l'état Bloqué - Inactif à un autre état ou quand un processus quelconque devient inactif.

Cette allocation se fait à la condition qu'il y ait assez de place disponible en M.C. pour le nombre de pages dont le processus a besoin pour s'exécuter (WSS) et bien sûr que le nombre de processus dans le "balance set" ne dépasse pas le maximum fixé (50).
Pour que le processus "i" puisse entrer dans le "balance set", les deux relations suivantes doivent être respectées.

(1) $\text{WSS proc}_i + \sum_{j \in \text{BALSET}} \text{WSS proc}_j \leq$ taille maximum disponible pour les processus utilisés

(2) nbre de proc $\in$ BALSET < 50

Un processus sort dynamiquement et volontairement du balance set lorsqu'on effectue une action ou attend un événement qui le fait devenir inactif.

Périodiquement (toutes les secondes) le système ajuste le "balance set" : il force certains processus à en sortir et d'autres, peuvent prendre leur place. Il existe deux raisons qui obligent un processus à sortir du "balance set".

1) A chaque processus est associé un "Balset Hold Quantum (B.H.Q.)" qui vaut WSS x 20 millisecondes de CPU. Ce "quantum" représente le temps maximum de CPU qu'un processus peut utiliser dans le "balance set". Si pendant l'ajustement du "balance set", le système repère un ou des processus dont le temps d'utilisation du CPU est supérieur à ce quantum, il le fait sortir.
   (Cette méthode est comme dans la littérature sous le nom de "tourniquet" ou "Round Robin Methad").
2) Il se peut qu'un déroulement anormal des routines de gestion de la mémoire centrale provoque une situation telle que, à un moment donné, la somme des pages utilisées pour chaque processus appartenant au "balance set" soit supérieure au nombre de pages disponibles pour l'utilisateur.
   Autrement dit :

   (3) $\sum_{j \in \text{BALSET}} \text{CWSS proc}_j >$ taille maximum disponible pour l'utilisateur

Dans ce cas, pour rétablir l'inégalité dans l'autre sens, il faudra faire sortir certains processus du "balance set".
Le système termine l'ajustement en parcourant la GO-LISTE et en déterminant s'il peut faire entrer dans le "balance set" certains processus ne s'y trouvant pas encore, tout en respectant les deux relations (1) et (2).

I.4. L'ALLOCATION DU CPU [DIG5]

Quand un processus a obtenu sa place en mémoire centrale, il ne lui reste plus qu'à obtenir le CPU. Pour choisir ce processus courant, le système parcourt la GO-LISTE dans l'ordre des priorités et sélectionne le premier processus du "balance set" qui n'est pas bloqué, donc le premier processus dans l'état Prêt - Actif.
Le processus perd le "CPU" quand il se bloque ou devient inactif.
De plus, on y applique la méthode du tourniquet ("Round Robin Method") par "quantum" de 200 msec. Quand le processus a utilisé pendant 200 msec. le processeur, il cède sa place à un autre et retrouve sa place dans le "balance set". Si le système ne trouve dans la GO-LIST aucun processus répondant aux conditions d'exécution ($\in$ à BALSET et non bloqué), il exécute le "nul-job".

C'est le scheduler qui décide quel programme à exécuter. C'est donc lui qui effectue cette tâche d'allocation du CPU, sans oublier celles précédemment décrites de gestion des processus (I.2.) et de gestion du "balance set" (I.3.2.).

I.5 GESTION DES ENTREES/SORTIES SUR DISQUES ET BANDES [DIG5, XIN]

Dans cette partie, nous allons décrire comment se passent les entrées/sorties courtes, c'est-à-dire sur disques et bandes, par opposition aux E/S sur terminaux, imprimantes, ... qu'elles sont longues.
Plusieurs processus peuvent demander ces ressources quels processus faut-il choisir ?

Nous commencerons pas une brève description des listes et blocs qui permettent de gérer ces E/S (I.5.1.) ensuite nous montrerons comment sont traitées ces E/S quant il s'agit d'un disque (I.5.2.) et quand il s'agit d'une bande (I.5.2.).

Toutes ces tâches sont effectuées par le module "PHYSIO" du système, qui gère les Entrées/Sorties physiques des processus.

I.5.1. Organisation des informations

Il existe des blocs de données pour chaque unité, chaque contrôleur, chaque canal. Ces informations étant décrites dans l'annexe A, nous ne les approfondirons pas ici. Nous signalerons seulement que la structure de ces blocs correspond à la configuration physique. C'est-à-dire que chaque disque est relié à un canal, plusieurs disques peuvent être reliés au même canal et il peut exister plusieurs canaux. Que chaque bande est reliée à un contrôleur et chaque contrôleur à un canal. Plusieurs bandes peuvent être reliées à un contrôleur de même que plusieurs contrôleurs à un même canal. Il peut exister plusieurs canaux et plusieurs contrôleurs, cela dépend de la configuration et surtout du nombre de bandes. Nous ajouterons également que toute information de ou vers les unités doit passer par le canal et que le canal est capable de gérer des informations pour positionner plusieurs unités en même temps, mais il ne peut transférer des pages d'informations que pour une seule unité à la fois.

Par contre une unité ne peut traiter qu'une demande à la fois.

Les demandes sont enregistrées dans des IORB's (I/O Request Blocks décrits dans l'annexe A) et à chaque unité sont associées deux listes de ces blocs.

PWQ : Position Wait Queue.
Liste d'attente du positionnement.

TWQ : Transfert Wait Queue.
Liste d'attente de transfert d'une page.

Nous concluons donc que le traitement des E/S sur disques et bandes exige l'enregistrement des demandes dans les deux files d'attentes PWQ et TWQ et une gestion vigilante de l'accès aux canaux et aux unités.

I.5.2. Le traitement des Entrées/Sorties sur disques

Sur un disque, c'est le temps de **positionnement** du bras qui est le point le plus critique, il requiert un temps non négligeable. Le système essaye donc de minimiser ce temps. Pour cela, il utilise l'algorithme d'optimisation connu dans la littérature sous le nom de "SCAN algorithme".

Avec cet algorithme, on ne choisit pas la première demande de l'unité (FIFO) mais bien un cylindre. On sélectionne dans la PWQ de l'unité, le cylindre pour lequel le déplacement du bras est le plus court, tout en donnant la préférence aux demandes d'écriture. En favorisant l'écriture, on tend à réduire le temps d'attente des processus dû au "swapping" car la lecture d'une page est généralement précédée par l'écriture sur disque d'une autre page.

Ce cylindre est cherché par rapport au cylindre courant.

Le premier cylindre de n° directement supérieur pour lequel il existe une demande d'écriture.

S'il n'y en a pas

le premier cylindre de n° directement supérieur pour lequel il existe une demande de lecture.

S'il n'y en a pas

le premier cylindre de n° directement inférieur pour lequel il existe une demande d'écriture.

S'il n'y en a pas

le premier cylindre de n° directement inférieur pour lequel il existe une demande de lecture.

Quand un cylindre est sélectionné et que le positionnement a été effectué, toutes les demandes concernant ce même cylindre sont transférées de la PWQ à la TWQ.

Toutefois, avec ce système, une demande risque d'attendre très longtemps dans la PWQ avant d'être servie. Pour éviter cet inconvénient, on utilise des "compteurs de fiabilité" ("fairness counts"). A chaque unité est associé un tel compteur. Grâce à lui, on autorise un nombre maximum de transferts (E/S) consécutifs suivant l'algorithme d'optimisation "SCAN" de sélection du positionnement. Quand ce maximum est atteint, on sélectionne le positionnement de la première demande de la PWQ, donc de la demande la plus ancienne. Cela implique également un nombre maximum de transferts consécutifs pour un même cylindre, c'est-à-dire un nombre maximum de demandes à faire passer de la PWQ dans la TWQ lorsqu'un cylindre est sélectionné.

Grâce à cet algorithme "SCAN", quand il se produit une demande d'E/S sur une unité et que l'unité est positionnée sur le cylindre de la demande, alors, on peut directement ajouter cette demande dans la TWQ de l'unité (sous réserve évidemment du "compteur de fiabilité de l'unité").

L'allocation du canal aux différentes unités, s'effectue en choisissant parmi toutes les unités positionnées associées au canal, l'unité dont la TWQ possède la demande la plus ancienne. Sous réserve cependant, d'un deuxième type de "compteur de fiabilité". Un tel compteur associé à chaque canal empêche qu'une unité de ce canal soit favorisée par rapport aux autres. Quand le nombre de transferts consécutifs pour une unité atteind le maximum, spécifié par le compteur, on sélectionne une autre unité. Le cas où une unité serait trop favorisée se produit quand il y a beaucoup d'E/S pour un même cylindre de cette unité. Toutes les demandes sont alors transférées en même temps dans le TWQ. Si par après, une demande arrive dans le TWQ d'une autre unité, cette dernière devra passer après toutes les précédentes : c'est le rôle du compteur de fiabilité du canal d'empêcher cela.

I.5.3. Le traitement des Entrées/Sorties sur bandes

La prise en charge d'une demande d'E/S sur bande est plus simple. En effet, une bande est de caractère séquentiel et, nécessite un rebobinnage avant le positionnement. Les demandes seront servies en FIFO et mises en files comme suit :

1. si la demande exige uniquement un transfert et que la PWQ est vide, elle est transférée dans la TWQ.
2. si la demande exige un positionnement, elle est ajoutée à la PWQ.
3. si la demande est un transfert et que la PWQ n'est pas vide, alors elle est ajoutée à la PWQ.

L'allocation du canal aux unités s'effectue par la méthode du tourniquet (Round Robin) c'est-à-dire,que le canal est alloué à chaque unité l'une après l'autre pour toutes les demandes de leur TWQ. Quand toutes les demandes de la TWQ d'une unité ont été traitées, le canal est alloué à l'unité suivante.

I.6. LES OPTIONS DE CONTROLE [DIG5]

Nous venons de décrire un système général et standard, mais certaines situations ne correspondent pas exactement à ce système. Pour adapter le système d'exploitation de TOPS-20 aux besoins individuels de chaque installation, il existe un certain nombre de paramètres d'adaptation. Nous allons les décrire rapidement et signaler quelles options sont actuellement en vigueur sur le système que nous étudions.

I.6.1. Le biais de contrôle

TOPS-20 tel que nous venons de le décrire, manipule des processus dans un contexte où la charge est composée de processus interactifs et de processus "compute-bound" avec cependant une majorité d'interactifs. Pour pemettre d'adapter le système à des charges moins typiques, TOPS-20 fournit le biais de contrôle.
Ce biais peut prendre des valeurs s'étendant de 4 à 20, la valeur par défaut étant 11.

Nous constatons qu'une valeur basse du biais favorise les processus interactifs, tandis qu'une grande valeur favorise les processus "compute-band".

Sur le système de l'institut, on adopte une valeur basse pendant la journée (1 ou 3), et une valeur élevée la nuit et le week-end (20) ; le système étant presque uniquement interactif la journée, tandis que les processus "compute-bound", les travaux en lots ("batch jobs") s'exécutent surtout la nuit et le week-end.
Nous prendrons le système en compte pendant la journée. A ce moment, la charge est essentiellement composée d'étudiants et de chercheurs, formant ainsi la charge universitaire. C'est donc à un système où les processus interactifs sont favorisés que nous nous intéressons.

I.6.2. L'établissement de classes ("class-scheduling")

Ce paramètre permet de diviser les utilisateurs en classes et d'allouer un pourcentage d'utilisation du "CPU" à chaque classe.

Pour cela, trois choses entrent en jeu : le nombre de classes (max. 8), les membres de chaque classe, et le pourcentage de CPU alloué à chaque classe. Dans une classe, ce poucentage est partagé équitablement entre chaque utilisateur. Un "job", et non chaque processus puis individuellement, appartient globalement à une classe.

La priorité d'un processus dans la GO-LISTE ne dépend plus uniquement des files auxiliaires, mais également de l'utilisation qu'il a déjà faite de la part qui lui a été attribuée : si il l'a très peu utilisée, sa priorité est élevée, dans le cas contraire, elle est basse.

Sur le système de l'institut, 4 classes sont établies avec des pourcentages de CPU différents selon la période de la journée.

classe	membres	jour pendant la semaine 8 h. à 19 h 45	Parfois	nuit et W.E.
classe 0	utilisateurs normaux	75 %	65 %	40 %
classe 1	administration	10 %	20 %	10 %
classe 2	centre	10 %	10 %	10 %
classe 3	batchs	5 %	10 %	40 %

La journée, les interactifs sont favorisés par rapport aux "batchs" le week-end et la nuit, cela se renverse. Les étudiants sont également très favorisés la journée.

I.6.3. L'utilisation des files spéciales

Le biais de contrôle et l'établissement de classes peuvent d'une certaine manière modifier le mouvement entre les files auxiliaires. Mais, il existe encore d'autres moyens pour poser des restrictions sur ce mouvement.

- le moniteur peut attribuer au "job" d'un utilisateur une garantie d'utilisation du CPU par un pourcentage. Les processus faisant partie de tels jobs sont placés dans la file auxiliaire 0, pour être sûr qu'ils soient favorisés par rapport à tous les autres. Quand le processus a dépassé le pourcentage de garantie qui lui a été attribué, il est envoyé dans une file "compute-bound", c'est-à-dire dans une des deux files 3 et 4.
  Ce paramètre ne peut pas être utilisé si l'on établit des classes.
- grâce à un appel moniteur, certains utilisateurs privilégiés peuvent spécifier une file minimum et une file maximum que ses processus ne pourront jamais dépasser.
- dans les conditions normales, il n'y a pas de processus dans la file 5. Mais on peut obliger les travaux en lots (batch jobs) à s'exécuter dans cette file.

Il faut remarquer que les deux premières possibilités affectent tous les processus du système et que la dernière peut obliger les "jobs" en lots à rester longtemps dans le système avant de se terminer.

Ces options ont été expérimentées par le centre de calcul mais n'ont pas été retenues.

I.6.4. "Prétransférer le "working set"

Quand un processus entre dans le "balance set", son "working set", est transféré en mémoire au fur et à mesure des demandes. TOPS-20 permet de stipuler qu'il soit transféré en entier et directement lorsque le processus est sélectionné.

Cependant, il faut constater que cela entraîne la création d'un certain nombre de demandes d'E/S, ce qui augmente soudainement la charge du disque.

A l'institut, le centre de calcul a testé les deux possibilités et n'a pas constaté de différence marquante.

I.7. PRECISION DES ETATS D'UN PROCESSUS PAR RAPPORT AU SYSTEME DECRIT ET TRANSITIONS ENTRE CES ETATS [VER]

Maintenant que nous avons décrit les mécanismes principaux du système, nous sommes capables par rapport à ce système de préciser ce que sont en pratique les différents états des processus (I.7.1.) qui ont été vus précédemment (cfr. I.1.4.). Nous pouvons aussi examiner les transitions entre ces états et leurs significations dans le système (I.7.2.).

I.7.1. Les états des processus dans le système décrit

Rappelons qu'un processus peut être bloqué ou prêt, actif ou inactif, et courant, et que les états sont des combinaisons de ces 3 types de situations.

Il y a cinq états :

- Bloqué - Actif (BA),
- Prêt - Actif (PA),
- Bloqué - Inactif (BI),
- Prêt - Inactif (PI),
- Courant (C).

Et voici comment nous pouvons les interpréter par rapport aux mécanismes que nous venons de voir.

BA : c'est un processus qui se trouve dans le "balance set" mais qui est bloqué. Or pour toute autre raison qu'une Entrée/Sortie courte, c'est-à-dire sur bande ou disque, le processus qui se bloque pert sa part de mémoire, il sort du "balance set". Un processus bloqué-actif est donc un processus du "balance set" bloqué pour une Entrée/Sortie courte.

PA : c'est un processus du "balance set" qui peut être sélectionné pour devenir courant.

BI : c'est un processus qui ne se trouve pas dans le "balance set" et qui est bloqué pour une raison autre qu'une Entrée/Sortie courte, c'est-à-dire pour une raison qui le classe dans une liste d'attente : E/S longue (sur terminal), terminaison d'un processus inférieur ... (cfr. I.2.2.). Un processus Bloqué - Inactif se trouve dans une liste d'attente (WTLST, TTILST, ...).

PI : c'est un processus qui ne se trouve pas dans le "balance set" mais qui n'est pas bloqué. Un processus Prêt - inactif est un processus qui voudrait entrer en mémoire centrale et qui attend d'avoir ce droit.

C : c'est le processus auquel le processeur est alloué.

I.7.2. Etats et listes

La relation qui existe entre les états d'un processus et les listes de processus est synthétisé par le diagramme de la figure I.5.

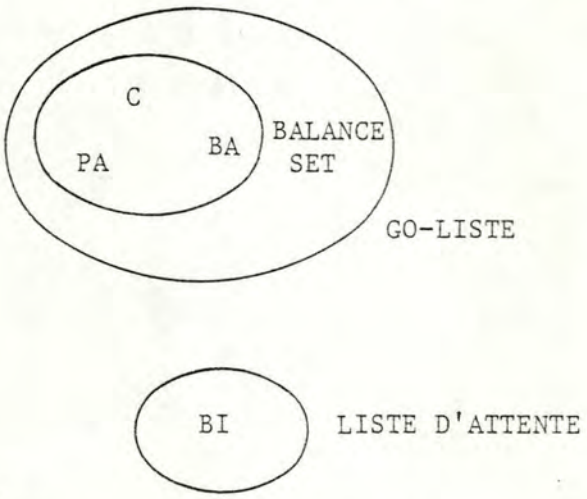


figure I.5.

- Les processus courants et actifs se trouvent dans le "balance set" et donc dans la GO-LISTE.
- Les processus Prêts - Inactifs se trouvent dans la GO-LISTE, mais pas dans le "balance-set".
- Les processus Bloqués - Actifs ne se trouvent pas dans la GO-LISTE, ils se trouvent dans les listes d'attente.

I.7.3. Les transitions entre les états

A partir de la signification qui vient d'être donnée des différents états, on peut par le schéma de la figure I.6. donner les transitions possibles entre chacun d'eux.

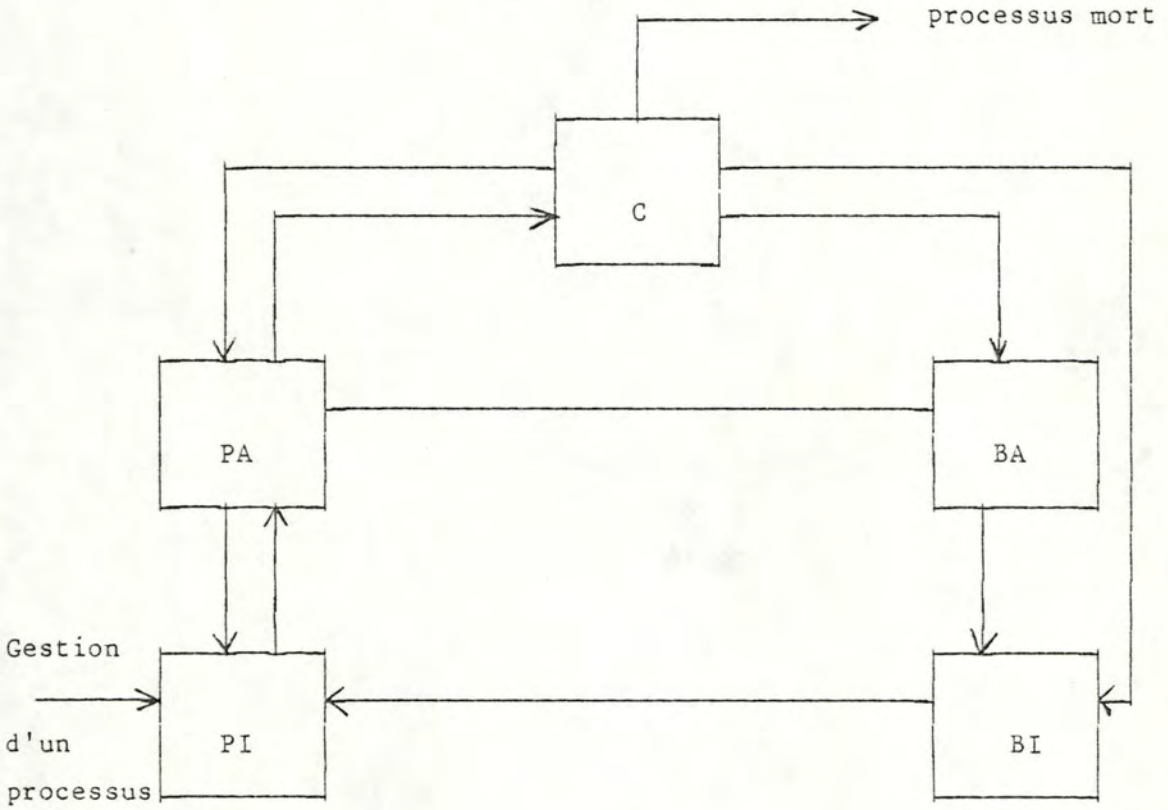


figure I.6.

-> **PI** : A sa création, un processus devient PI, il n'est pas bloqué, mais ne se trouve pas dans le "balance set" et demande d'y être.

PI -> PA : Un processus est sélectionné par le " scheduler" pour entrer dans le "balance set". Seul un processus Prêt - Inactif peut devenir actif.

PA -> PI : Lors de l'ajustement du "balance set" un processus est forcé d'en sortir.

PA -> C : Un processus est sélectionné pour être courant. Seul un processus PA peut être choisi pour s'exécuter.

C ->PA : Le processus courant perd le processeur parce qu'il s'est exécuté durant 200 ms. Il a terminé son "quantum" : il reste dans le "balance set" et ne se bloque pas.

C ->BA : Le processus courant perd le processeur parce qu'il veut effectuer une Entrée/Sortie sur disques ou bandes : il se bloque mais reste dans le "balance set".

C ->BI : Le processus courant perd le processeur pour une autre raison qu'une Entrée/Sortie courte.
Il se bloque, sort du "balance set" et entre dans une liste d'attente.

BA -> PA : La raison pour laquelle le processus était bloqué est supprimée (fin d'E/S courte), il devient prêt et reste dans le "balance set".

BI -> PI : La raison pour laquelle le processus était bloqué est supprimée, il sort de sa liste d'attente et devient candidat pour entrer dans le "balance set".

C -> : Le processus termine son exécution, il termine sa vie.

CHAPITRE II

MODELISATION DU SYSTEME ET MESURES NECESSAIRES

II.0. INTRODUCTION

Dans le chapitre I, nous avons décrit de manière assez détaillée le système d'exploitation qui gère l'ordinateur DEC-20 installé à notre institut. Nous avons vu, qu'il existait une compétition entre les différents processus du système quant à l'allocation des différentes ressources (unité centrale de traitement, mémoire, canaux, ...). Il s'en suivait obligatoirement la création de files d'attente de ces processus en vue de l'obtention de ces ressources, ce qui ouvre la voie à une modélisation du système. Mais avant d'en venir à la manière dont elle a été réalisée, nous allons d'abord nous attarder sur ce concept de modélisation. Que représente-t-il ? A quoi peut servir un modèle ? (II.1.) - Nous exposerons alors la modélisation du système que nous avons adoptée et essaierons de la justifier (II.2.). Nous partitionnerons les processus en classes de manière à "raffiner" un peu plus le modèle (II.3.).

Nous donnerons ensuite les différentes mesures nécessaires à effectuer sur le système réel (II.4.) avant d'en terminer avec quelques remarques générales (II.5.).

II.1. DEFINITION ET UTILITE D'UN MODELE [DEBU]

II.1.1. Définition

Un modèle est une abstraction du réel : il reproduit une situation réelle de manière fort simplifiée mais cependant, en en gardant les propriétés fondamentales qui régissent son comportement de telle sorte, qu'un profil du comportement réel puisse être donné grâce à l'analyse du modèle. Dans le cas présent, nous allons donc essayer de modéliser la manière dont le système d'exploitation gère la concurrence entre les différentes requêtes des processus.

II.1.2. Validation d'un modèle

Le modèle étant établi, il faudra bien sûr qu'il soit valable pour qu'on puisse l'utiliser par la suite. La figure II.1. donne un schéma type de validation de différents types de modèles. On y voit les différentes étapes à suivre.

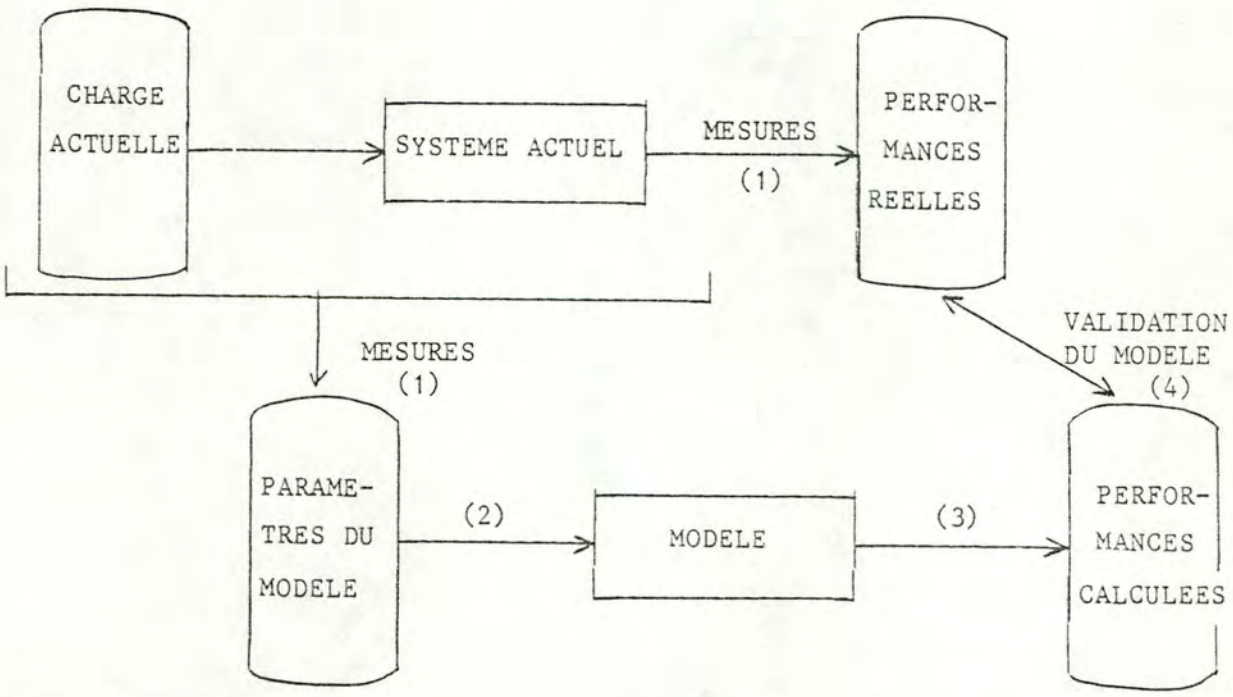


fig. II.1. : mesure des paramètres et validation d'un modèle

Premièrement, sur le système actuel "tournant" normalement sous une certaine charge (la charge actuelle), on effectue certaines mesures de performances réelles (pourcentages d'utilisation des différents serveurs du modèle p. ex.). On mesure alors aussi certaines valeurs concernant les appareils ainsi que la charge du système. Celles-ci constituent les paramètres du modèle.

Ensuite, on peut appliquer le modèle à ces paramètres, et comparer les résultats fournis (performances calculées) avec les valeurs effectivement mesurées (performances réelles).
Nous donnons respectivement en II.4.1. et II.4.2. les performances réelles à mesurer ainsi que les paramètres du modèle. Le chapitre III sera consacré à la mise en oeuvre de ces mesures. Dans le chapitre IV , nous appliquerons les paramètres au modèle dont nous testerons alors la validité.

II.1.3. Rôle prévisionnel d'un modèle

Si sur plusieurs périodes d'observations différentes, les valeurs calculées par le modèle correspondent plus ou moins aux valeurs effectives du système, alors, on peut être amené à croire que le modèle est bon.
On pourra alors ensuite l'utiliser en toute confiance pour prédire le comportement futur et ainsi évaluer certains changements proposés dans le système. Il suffira en effet, conformément au schéma de la figure II.2., d'ajuster les paramètres du modèle qui seront modifiés lors de la période dite "de projection" (parfois d'ajuster le modèle), de faire "tourner" le nouveau modèle avec nouveaux paramètres, ce qui fournira les performances estimées pour la période projetée.

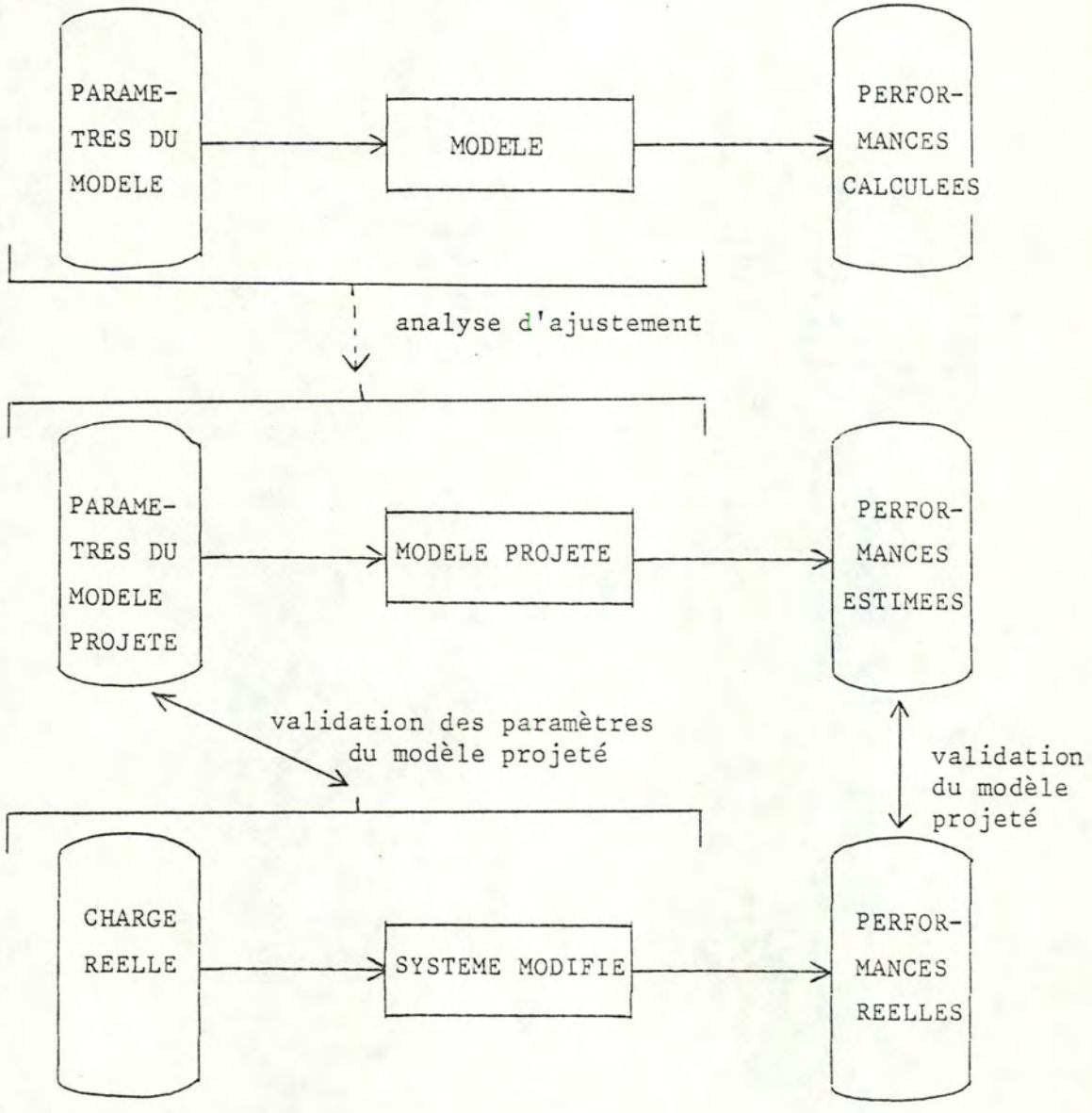


fig. II.2. : prédiction.

Plus tard, lorsque cette période aura été atteinte, on pourra alors de nouveau valider le modèle en comparant les performances estimées avec les performances réelles qu'on aura mesurées sur le système modifié.

II.2. MODELISATION DU SYSTEME

II.2.1. Le modèle

Nous n'exposerons pas ici les différents outils de modélisation possibles ni l'outil choisi. Tout cela est expliqué en détail dans le chapitre IV ainsi que dans l'annexe concernant le logiciel QNAP. Venons en plutôt à la modélisation proprement dite.

La principale difficulté rencontrée lors de l'établissement du modèle était due au fait que le modèle doit représenter l'acheminement séquentiel, dans le temps et dans l'espace, des différents processus du système. Cela est rendu difficile, à partir du moment où l'on sait que les processus peuvent utiliser plusieurs ressources à la fois (ex. : le "CPU" et la mémoire centrale ; le canal et un disque). La figure II.3. constitue le modèle tel que nous l'avons défini.

Nous avons choisi de représenter le modèle sous forme d'un graphe orienté car il s'agit de la manière la plus simple de représenter un réseau de files d'attente. Les rectangles représentent les serveurs et les arcs indiquent les différents chemins que les processus peuvent suivre lorsqu'ils se déplacent dans le réseau. Devant chaque serveur se trouve la file d'attente des processus attendant d'être servis. Le triangle (Δ) n'est pas un serveur au sens propre : il bloque les processus, seul un nombre déterminé de processus sont autorisés à le traverser. Le triangle renversé (∇) quant à lui, débloque des processus : quand un processus passe à travers ce triangle renversé, un processus de plus peut traverser le triangle droit.

Ce modèle représente en fait le comportement d'un processus dans un univers de multiprogrammation : il est caractérisé par une alternance de périodes d'utilisation processeur et de périodes d'entrées/sorties, précédées en général par des attentes devant ces serveurs.

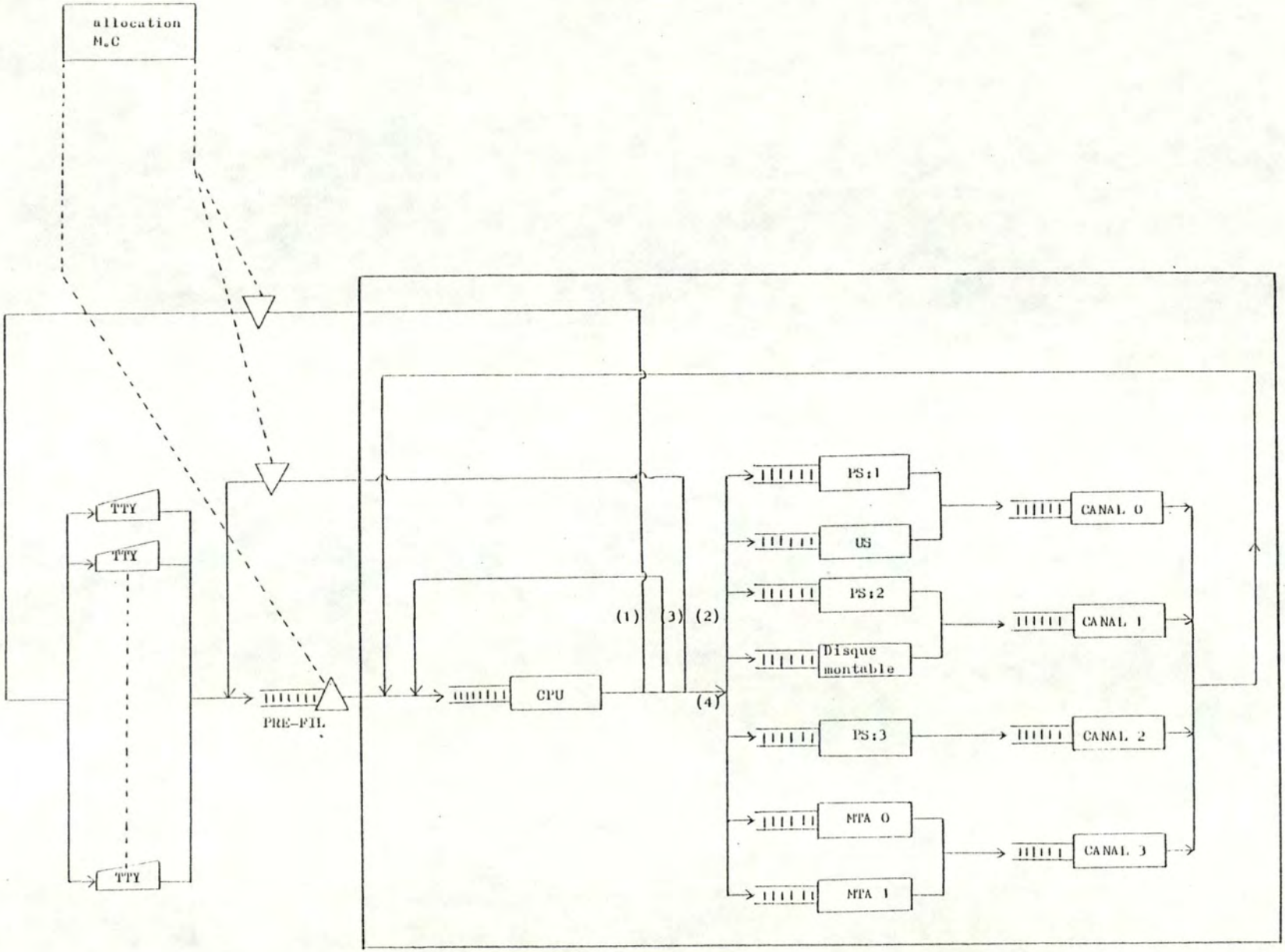


fig. II.3. Modèle du système

II.2.2. Description du modèle

II.2.2.1. Décomposition en deux niveaux

Pour plus de clareté, nous avons décomposé le modèle en deux niveaux : le niveau interne comprenant les processus appartenant au "balance set" et le niveau externe comprenant les processus ne lui appartenant pas.

1) le niveau externe est constitué

. des différents **serveurs** que sont les terminaux

. des différents **processus** effectuant des E/S à ces terminaux. Ces processus n'appartiennent pas à la GO-LIST ; ils font partie des processus Bloqués - Inactifs (BI) et appartiennent à l'une des listes d'attente du système (cfr. I.2.2.)

. de **"gestionnaires"** de l'accès au niveau interne (les triangles $\triangle$ $\triangledown$), c'est-à-dire de l'accès au "balance set". Représenter dans le modèle la gestion du "balance set" telle qu'elle se déroule en réalité serait très compliqué.
Pour la gestion dynamique (cfr. I.3.2.), il faudrait étudier la taille du "working set" de tous les processus. Quant à la gestion périodique, on peut modéliser le BHQ, nous en parlerons plus tard. En ce qui concerne l'autre raison qui force un processus à sortir du balance set, elle est très difficile à modéliser.
Aussi, avons-nous considéré le nombre moyen de processus dans le "balance set" pour déterminer le nombre de processus dans le niveau interne du modèle.
Un processus est autorisé à entrer dans le niveau interne si le nombre de processus s'y trouvant déjà est inférieur à ce nombre moyen. Quand un processus sort du niveau interne, il permet à un autre processus de prendre sa place (représenté par le triangle renversé).

. de la file d'allocation de la mémoire centrale (file devant) que nous appellerons par la suite PRE-FIL. Cette file est constituée des différents processus de la GO-LIST non bloqués et qui n'appartiennent pas au "balance set" (c'est-à-dire qui n'ont pas leur "working set" en mémoire centrale). Il s'agit donc des processus Prêts - Inactifs (PI). Ces processus sont prêts à entrer dans le niveau interne quand il y aura de la place en mémoire centrale. Ils sont ordonnés par priorités (cfr. I.2.2.).

Les processus se trouvant dans le niveau externe sont tous inactifs (PI ou BI). Il s'en suit donc, qu'aucun de ces processus n'appartient au "balance set".

2) le niveau interne est constitué

. d'une file comprenant les processus appartenant au "balance set" et qui n'attendent pas une entrée/sortie sur disque ou bande. Ceux-ci sont prêts à utiliser le "CPU". Il s'agit des processus Prêts - Actifs (PA). Ils sont également ordonnés par priorité.

. des différentes files d'attente devant les unités disques et bandes (pour le positionnement) ainsi que devant les canaux (pour le transfert). Les processus qui les forment appartiennent aussi au "balance set". Ce sont les processus Bloqués - Actifs (BA).

On peut donc remarquer que l'ensemble des processus qui se trouvent dans le niveau interne constitue en fait le "balance set".

Il faut aussi ajouter que pour nous, la PWQ est la file devant chaque disque et la TWQ la file devant chaque canal. En réalité, il y a une PWQ et une TWQ par unité. De cette manière, nous aurions dû avoir plusieurs files devant chaque canal, et le canal traitant la file possédant la demande la plus ancienne.

Or cela n'est pas possible dans un modèle. Nous avons donc une seule file par canal et lorsque le canal traitera la demande d'une unité, il devra traiter toutes les commandes présentes dans la file pour cette unité.

. des **serveurs** que sont le "CPU" ainsi que les différentes unités (positionnements et transferts) et les canaux (transferts).

Il faut souligner ici que lors d'un transfert, il y aura toujours deux ressources occupées : en effet, le "serveur" disque restera occupé par la même demande d'E/S pour laquelle le canal est en train d'assurer le transfert de données.

II.2.2.2. Transitions lors de la sortie du CPU

En ce qui concerne le routage des différents processus dans le réseau, il faut différencier les différentes sorties du "CPU" ; elles sont de 5 types : (le numéro entre paranthèse se réfère au numéro sur la figure II.3.).

(1) "CPU" vers les terminaux : un processus quitte le "balance set" et va dans une des listes d'attente (voir en I.2.2.) pour effectuer une sortie ou une entrée sur un terminal. Pour en revenir aux transitions entre états vues en I.7.3., nous pouvons dire qu'il s'agit d'une transition C → BI. Dans ce premier cas, nous avons émis l'hypothèse d'une continuité dans l'occupation des terminaux : si un utilisateur a fini son travail ("job"), un nouvel utilisateur prend immédiatement sa place si bien que dans le modèle, on ne voit aucune différence.

(2) "CPU" vers la PRE-FIL (file d'allocation de la mémoire) : quand un processus a épuisé sa tranche de temps (BHQ : cfr. I.3.2.) d'utilisation "CPU", ce processus quitte le "balance set" (transition PA → PI).

Bien sûr, s'il n'y a aucun autre processus PI, celui-ci réentrera directement dans le "balance set" avec son "BHQ" réinitialisé.

Rem. : Pour simplifier le modèle, nous avons placé cette transition à la sortie du "CPU". En réalité, il y a un test périodique du "Balance set Hold Quantum" (I.3.2.) de tous les processus du "balance set", ce qui veut dire que ce ne sera pas nécessairement celui qui occupait le "CPU" qui devra sortir du "balance set".

(3) "CPU" vers la file "CPU" : quand un processus a épuisé sa tranche de temps de 200 millisecondes d'utilisation du "CPU", il perd le "CPU" et retourne dans la file d'attente du "CPU" (transition C → PA). Bien sûr, s'il n'y a aucun autre processus PA, celui-ci réutilisera directement le "CPU" avec sa tranche de temps réinitialisée.

Rem. : Il ne faut pas oublier que toutes les 100 millisecondes, la GO-LIST est réordonnée selon les niveaux de priorité des "forks", ceci pour préciser que les processus réentrant respectivement dans les deux derniers cas au début de la PRE-FIL et de la file "CPU" retrouveront rapidement dans la file une place conforme à leur priorité.

(4) - "CPU" vers les unités disques : cas typique du défaut de page : le processus a besoin d'une ou de plusieurs pages se trouvant sur disque et non en mémoire centrale. Dans l'autre sens, il y a bien sûr le transfert (sauvetage) de pages de mémoire centrale vers les disques.

- "CPU" vers les bandes magnétiques : cas du "back up" (sauvetage de fichiers sur bandes) ou de demande de lecture sur bande.

Dans ces deux derniers cas, nous avons donc affaire à la transition C →BA : les processus sont bloqués mais ils restent dans le "balance set".

II.2.3. Justification du modèle

Nous allons essayer de justifier notre modèle par rapport à toutes les transitions entre états possibles dans le système (cfr. I.7.3.) L'idéal serait qu'à chacune de ces transitions d'état corresponde une transition dans notre modèle. Mais par simplification, elles ne sont pas toutes reprises. Nous justifierons donc les cas de transitions d'état non prises en compte explicitement par le modèle.

Dans le point précédent (II.2.2.), nous avons expliqué le modèle, et particulièrement (II.2.2.2.) les transitions à la sortie du CPU. Nous avons vu que ces transitions de notre modèle correspondaient à des transitions dans le système. Rappelons-les :

- la transition (1) correspond au changement d'état C → BI pour des E/S sur terminal.
- la transition (2) correspond au changement d'état PA → PI dans le cas où le processus a épuisé son "Balance set Hold Quantum" (BHQ).
- la transition (3) correspond au changement d'état C → PA
- la transition (4) correspond au changement d'état C → BA

Les autres changements d'états pris en compte par notre modèle sont les suivants :

- PI → PA : Un processus passe du niveau externe au niveau interne. Il entre dans le "balance set". Il y a maintenant suffisamment de place en mémoire centrale pour son "working Set", car un autre processus est sorti du niveau interne (C -> BI ou PA -> PI) (cfr. I.3.2.).

- PA → C : Le processus PA se trouvait dans la file d'attente devant le "CPU". Il vient d'être sélectionné pour être courant. Il occupe maintenant le "CPU"
- BA → PA : Dans le modèle, le processus sort d'un des canaux et se dirige vers le début de la file d'attente devant le "CPU".
  Une entrée/sortie courte sur disque ou bande vient de se terminer, le processus se débloque et redevient Prêt à utiliser le "CPU".
- BI → PI : Une E/S sur terminal vient de se terminer. Le processus quitte un des serveurs "TTYS" et entre dans la file d'attente pour l'allocation de la mémoire. Il se débloque donc mais reste inactif (reste dans le niveau externe). En ce qui concerne les processus BI, uniquement les processus en attente d'E/S sur terminal ont été pris en compte explicitement dans le modèle.

Justifions maintenant pourquoi le modèle ne s'occupe pas des transitions d'états suivantes :

- → PI : création d'un processus.
- C → : mort d'un processus.
- C → BI : entrée dans une liste d'attente mais pour une autre cause qu'un début d'E/S sur terminal.
- BI → PI : sortie d'une liste d'attente mais pour une autre cause qu'une fin d'E/S sur terminal.
- PA → PI : perte de l'accès à la mémoire centrale, mais pour un autre cas que le BHQ épuisé.(cfr. I.3.2. (3)).

Nous avons choisi de considérer un réseau fermé : il y a un nombre constant de processus qui "tournent" dans le modèle ; aucun n'y entre, aucun n'en sort. Cela explique que les deux transitions d'états : création et mort d'un processus ne sont pas considérées.

Nous estimons également qu'un processus appartenant à une liste d'attente autre que celles des entrées et sorties sur terminal peut être considéré comme un processus mort ou pas encore créé. En effet, un tel processus ne se manifeste plus au système avant un long laps de temps : il dort et n'utilise aucun serveur. Ainsi, de la même manière que pour la création et la mort d'un processus, les transitions d'états mettant ou retirant un processus d'une liste d'attente autre que celles pour une E/S (C → BI et BI -> PI) ne seront pas prises en compte dans le modèle.

Nous avons déjà expliqué en décrivant la décomposition de notre modèle en deux niveaux (cfr. II.2.2.1.), qu'il nous était impossible de modéliser la gestion entière du "balance set". Dès lors, nous avons considéré le nombre moyen de processus en mémoire centrale. Or la sortie d'un processus du "balance set" par la transition PA → PI est une des fonctions de la gestion du "balance set" (cfr. I.3.2. (périodiquement)).
Cette transition ne devrait donc pas être considérée du tout. Cependant, nous avons pris en compte le cas typique du "BHQ" épuisé car il découle, entre autre du temps passé dans le "CPU".

II.3. PARTITIONNEMENT EN CLASSES

Nous avons dit qu'un modèle jouait un rôle important pour la prévision du comportement futur d'un système. Aussi avons-nous jugé utile de partitionner les processus du système en différentes classes : en effet, un système assume diverses catégories de travaux qui consomment des ressources : travaux interactifs (en temps partagé), travaux en "batch" etc...

Il est donc important, trouvons nous, que le modèle donne des résultats différents par classes, de telle sorte que l'on puisse vérifier si chaque catégorie de travail consomme le niveau désiré des différentes ressources du système.

Grâce à la nature des différentes files auxiliaires du système (cfr. I.2.3.), il nous a été facile de diviser les processus du système en 6 classes :

- . les processus privilégiés correspondant à la file 0 (classe 0).
- . les processus très interactifs correspondant à la file 1 (classe 1).
- . les processus interactifs (mais à temps d'utilisation du CPU plus long) correspondant à la file 2 (classe 2).
- . les processus "mixtes" et "compute-bound" correspondant à la file 3 (classe 3).
- . les processus très "compute-bound" ou "batchs" (travaux en lots) correspondant à la file 4 (classe 4).
- . les processus à plus basse priorité correspondant à la file 5 (classe 5).

la première (file 0) et la dernière (file 5) classe sont moins importantes vu le faible nombre de processus présents dans les files 0 et 5.

Rem. : Cas des "Batch jobs"

Les travaux en lots ("batch") sont des "jobs" dont les données en entrée sont toutes contenues dans un fichier qu'on appelle le fichier de contrôle.

Le fichier de contrôle peut être soumis, par l'intermédiaire d'un utilisateur, à un terminal ou bien créé suite à la lecture d'un ensemble de cartes perforées. Le fichier de contrôle est ensuite transféré sur disque ("spooling"). Lors de l'exécution des travaux en lots, les caractères du fichier de contrôle sont traités comme si ils étaient introduits à un terminal réel. Dans le comportement réel, la lecture se fait sur disque ; il n'y a donc pas d'interactions avec les terminaux, c'est pourquoi les travaux en lots deviennent rapidement "compute-bound" (file 4) et le restent.

Dans le modèle, nous n'avons pas établi de réseau ouvert spécialement destiné à la représentation des processus "batchs" soumis au système par l'intermédiaire d'un lecteur de cartes. Et ceci pour deux raisons : d'une part, à cause de la faible importance qu'occupe ce type de travaux dans le système en temps normal (voir en I.6. les "control options" choisies pour les "batchs"); d'autre part, à cause de la complexité que cette situation apporterait au modèle.
En effet, si le système est ouvert, le nombre total de processus dans le système variera selon que des processus entreront ou quitteront le système. De plus, on devra fixer les taux d'entrée et de sortie ("troughput") du système, en ce qui concerne les processus "batchs". On obtiendrait ainsi un système mixte, c'est-à-dire comprenant un réseau ouvert pour les "batchs" et un réseau fermé pour les autres processus ; cela devient relativement complexe.

Nous dirons donc que les processus "batchs" sont en nombre fixe dans le système (le réseau est fermé), ils sont initialisés par les terminaux et ils correspondent aux processus les plus "compute-bound" (classe 4). Nous les assimilerons donc à ces derniers.

II.4. MESURES A EFFECTUER

Nous pouvons maintenant déterminer les différentes mesures que nous devons réaliser sur le système réel et qui correspondent aux paramètres nécessaires pour faire "tourner" le modèle ainsi qu'à certaines valeurs de validation du modèle.

II.4.1. Mesures de validation du modèle

Nous exposons ici les différentes mesures susceptibles d'être effectuées. Nous n'en avons effectué que certaines, les jugeant suffisantes pour notre modèle.

- "troughput" : taux de sortie par unité de temps.
  Il n'est pas important dans notre cas car le réseau considéré est fermé.

Nous avons jugé suffisant d'effectuer les mesures suivantes :

- % utilisation "CPU" : pourcentage de temps pendant lequel le "CPU" est utilisé.
- % utilisation des unités (PS:1, PS:2, PS:3, US, DSK MONT, MTA0, MTA1).
  N'oublions pas que, pour une même demande d'E/S, une unité est bloquée pendant le positionnement et le transfert par le canal. Le pourcentage de temps pendant lequel une unité est utilisée devra donc tenir compte,et du temps de positionnement de cette unité,et du temps de transfert pour cette unité.
- % utilisation des canaux (0, 1, 2 et 3) : pourcentage de temps pendant lequel un canal est affecté au transfert de données (pour n'importe quelle unité lui étant associée).
- % utilisation de chaque terminal : pourcentage de temps pendant lequel chaque terminal est en train de servir une Entrée et/ou une Sortie pour un processus du système.

Ce que nous avons voulu mesurer est donc le pourcentage de temps pendant lequel les différents "serveurs" de notre modèle sont utilisés par les processus du système.

II.4.2. Mesures des paramètres du modèle

Il y a deux types de paramètres que nous devons mesurer : ceux que nous devons connaître globalement pour toutes les classes et ceux que nous devons différencier par classes.

1) valeurs à mesurer pour toutes les classes globalement (ces valeurs ne dépendent donc pas de la nature du processus).
   - Le temps moyen de service des différents canaux (transfert). Dans notre modèle, ce temps de service sera donc la tranche de temps qu'un processus entrant dans un des canaux passera avant d'en sortir (temps du transfert d'une page).
   - Le temps moyen de positionnement des différentes unités. Dans notre modèle, il s'agira donc de la tranche de temps qu'un processus passera dans un "serveur" unité pour le positionnement (N'oublions pas que cette unité restera bloquée ensuite pour le transfert par le canal).
   - Le nombre moyen, pour un positionnement effectué, d'"input Output Request Blocks" transférés de la PWQ vers la TWQ. En effet, nous avons jugé utile de pouvoir tenir compte dans notre modèle du fait que (cfr. "SCAN" algorithme en I.5.2.), lorsqu'un positionnement a été effectué sur un cylindre, un certain nombre de demandes d'E/S concernant ce même cylindre peuvent être transférées directement de la PWQ vers la TWQ.
   - Le nombre moyen de processus dans le "balance set". Lors de l'initialisation du modèle, il s'agira donc du nombre de processus présents dans le niveau interne.

Pour les 2 premières mesures, il n'est pas nécessaire de les effectuer séparément par classe de processus, vu que les temps de positionnement et de transfert ne dépendent pas de la nature du processus mais bien de la charge du système ainsi que de la position sur l'unité de la ou des pages à transférer (par rapport à celle du bras de lecture) (cfr. I.B.2.).

2) valeurs à mesurer de manière distincte pour les différentes classes de processus, (car l'on se rend bien compte que ces valeurs varieront certainement selon que les processus appartiennent à telle ou à telle classe).

- temps moyen de service des terminaux : temps total entre le moment où un processus sort du "CPU" et se met en attente d'E/S terminal et le moment où il réentre dans la GO-LIST .

  Cette mesure s'appliquera aux processus interactifs.
- temps moyen de service du "CPU". Il s'agira dans notre modèle de la tranche de temps qui sera allouée chaque fois qu'un processus entrera dans le "CPU".
- les différentes probabilités de transition (routage des processus à la sortie du "CPU") (cfr. II.2.2.2.).
  - . "CPU" vers terminaux. Elle devrait être grande pour les processus interactifs (classes 1 et 2) et faible pour les processus "compute- bound".
  - . "CPU" vers début de la PRE-FIL (attention Rem. en II. 2.2.2. (2)).
  - . "CPU" vers la file du "CPU".
  - . "CPU" vers les différentes unités. En ce qui concerne les disques, cette valeur devrait être relativement grande, vu tous les mécanismes d'E/S courte pour gérer la mémoire centrale et les défauts de page.
- le nombre moyen de processus dans le modèle.

Lorsque nous utiliserons le modèle, nous devrons fixer le nombre de processus présents dans chaque classe (0 -> 5), de telle sorte que le modèle fournisse des résultats séparés par classe.
Cette mesure nécessitera de plus amples détails par la suite, (III.2.5.) car il ne faut pas oublier que les files auxiliaires contiennent tous les processus, qu'ils soient de la GO-LIST ou d'une des listes d'attente du système. Or, comme nous l'avons déjà dit en II.2.3., le modèle ne considère pas explicitement ces listes d'attente (excepté pour les E/S terminal).

Rem. : Certaines mesures s'ajouteront certainement par la suite aux mesures nécessaires énoncées, soit parce qu'elles ne sont pas encore apparues comme nécessaires à ce moment du travail, soit parce qu'elles résultent de la décomposition d'une mesure complexe à effectuer en plusieurs autres plus aisées.

II.5. REMARQUES GENERALES

Il va de soi que certains paramètres du modèle sont variables selon la charge du système : par exemple, si le "scheduler" minimise le mouvement du bras lors du positionnement (ce qui est le cas ici : "SCAN" algorithme), la mesure du temps moyen du "seek" durant une période légèrement chargée peut être significativement différente de celle effectuée lors d'une période fortement chargée. On devra donc faire des mesures à des moments différents, sous différentes charges du système. On insérera dans le modèle les paramètres mesurés dans les circonstances désirées et ainsi, si on veut voir le modèle s'appliquer au système fortement chargé, on lui fournira les paramètres mesurés dans de telles circonstances. Il est cependant important de noter que les résultats fournis pour un système fortement chargé seront les plus intéressants (et les plus exacts vu l'hypothèse de l'occupation constante de tous les terminaux).
On pourra aussi, à partir des différentes mesures sous les différentes charges, obtenir des fonctions analytiques pour les paramètres. Ceci nous évitera de devoir introduire des paramètres différents selon les charges différentes. Le seul paramètre à introduire serait alors la charge.

Tous les mécanismes vus au chapitre I correspondent en fait à des décisions internes du système visant avant tout à assurer un partage équilibré des ressources. Il ne faut pas oublier qu'il existe des variables de contrôle qui permettent aux responsables du système de prendre des "décisions externes" qui peuvent modifier fortement l'algorithme d'ordonnancement. Ceci afin de pouvoir adopter le comportement du système aux besoins du site. ex. : mettre les travaux en lots ("batch") dans la classe 5.
Il est donc important de tenir compte des options de contrôle qui ont été choisies (voir les "control options", au chapitre I.6.).

CHAPITRE III

LE MESUREUR ET SES RESULTATS

III.0. INTRODUCTION

Après avoir étudié le système et adopté une modélisation pour ce système, nous en avons déduit les mesures nécessaires pour réaliser le modèle. Nous sommes maintenant capables de construire le programme de ce mesureur.

Dans ce chapitre, nous allons décrire la manière dont nous avons réalisé le mesureur. Nous présenterons d'abord les outils dont nous disposons pour le réaliser et ceux que nous avons utilisés (III.1.).

Ensuite, nous décrirons de manière générale la réalisation du mesureur (III.2.). Pour les lecteurs plus intéressés par l'implémentation d'un tel mesureur, nous examinerons notre réalisation de manière plus approfondie (III.3.). Enfin, nous pourrons observer les mesures obtenues et les analyser (III.4.). Nous terminerons en parlant des perturbations introduites par le mesureur (III.5.). Nous signalons que le programme implémentant le mesureur et les résultats que nous avons obtenus se trouvent en annexe.

III.1. LES OUTILS DE MESURE [ADVE, CORN, DAWI, DIG3]

Avant de décrire les outils que nous avons utilisés pour construire notre mesureur, nous allons d'abord présenter quelques généralités concernant de tels outils (III.1.1.). Ensuite nous examinerons les outils à notre disposition et présenterons ceux que nous avons choisis (III.1.2.).

III.1.1. Généralités sur les outils de mesure

Un système informatique en exécution est, en fait, un ensemble d'objets dont les valeurs évoluent avec le temps et une suite d'événements correspondant aux changements d'état résultant de l'utilisation de ces objets . L'objectif d'un mesureur est de prélever le maximum d'informations sur le matériel, mais aussi et surtout sur le logiciel. C'est uniquement sur les outils permettant de capter des phénomènes logiques c'est-à-dire des phénomènes concernant le système d'exploitation, que nous nous pencherons ici.

On distingue deux techniques de mesures. D'une part, la mesure **sur événements** enregistre la situation de certaines variables au moment de l'occurence de certains événements. D'autre part, on peut effectuer un ensemble de mesures **par échantillonage** à intervalles réguliers. Ces deux manières de faire ont chacune leurs inconvénients. Quand on procède à une mesure sur événements, il est rarement possible de prévoir le nombre de fois que l'on passera par tel ou tel point de mesure et de plus, la mise au point (surtout la localisation des points de mesure) risque d'être difficile. Tandis que le problème principal lors de la mise en oeuvre d'une technique par échantillonage est de déterminer la période d'échantillonage. En effet, il faut arriver à établir un compromis entre la finesse des observations, le volume des données et le temps passé à effectuer les mesures.

Quelle que soit la technique utilisée, le mesureur doit pour le moins présenter certaines qualités :

1) La collecte de données ne doit pas introduire de perturbations, ou du moins le moins possible, dans le fonctionnement normal du système. En fait, il faudrait toujours s'assurer de ne pas trop modifier les performances du système.

2) L'accès aux variables à mesurer doit être garanti. La majorité de celles-ci sont en effet protégées à différents niveaux.

3) Le mesureur doit être adaptable et extensible.

4) Le mesureur doit être facile à mettre en oeuvre et à employer.

III.1.2. Les outils disponibles

Sur le système TOPS-20 implémenté au "centre de calcul" des facultés, il existe plusieurs outils permettant d'effectuer des mesures. Nous allons décrire ici ceux dont nous avons pris connaissance et expliquer les raisons qui nous ont fait préférer certains de ces outils à d'autres.

III.1.2.1. Outils permettant de mesurer le moment de l'occurence d'événements

Dans le système étudié, il existe plusieurs manières pour déterminer à quel moment un événement se produit :

La première manière est de saisir le temps de base du système qui est une valeur en nanosecondes se trouvant dans le "timer". Pour cela, on utilise l'instruction "DATAI" permettant de transférer en mémoire centrale des données lues sur n'importe quel appareil. Elle permet aussi de lire dans d'autres éléments définis comme des appareils internes du système et c'est ici qu'elle est intéressante pour nous, car le "timer" est assimilé à un tel appareil.

Cette instruction doit être utilisée de la manière suivante :

```
DATA    TIM, LOC
```

où TIM doit être défini comme le numéro identifiant le "timer"
et LOC doit être défini comme l'adresse mémoire où l'on veut transférer le temps de base du système.

Une autre manière de faire est d'utiliser les appels-systèmes TIME ou HPTIM qui respectivement fournissent le temps passé depuis que le système a démarré, et ce même temps mais, avec une précision beaucoup plus grande.

Cependant, ces deux moyens nous sont apparus assez complexes, alors que dans les données du système se trouve une simple variable (TODCLK) incrémentée de manière monotone et indiquant le temps en millisecondes. Cette variable peut être exploitée facilement par l'instruction suivante :

```
MOVE    AC, TODCLK
```

Il est évident qu'un appel-système consomme beaucoup plus de temps "CPU" qu'une simple instruction, nous écartons donc cette solution. D'autre part, il nous paraît suffisant de disposer du temps en millisecondes et de plus, c'est cette variable TODCLK que le système utilise pour mesurer le moment d'un événement ou calculer le temps passé entre deux événements.
Pour toutes ces raisons, c'est sur cette dernière solution (TODCLK) que notre choix s'est porté.

III.1.2.2. L'outil GETAB

L'outil GETAB est l'exemple même d'un outil permettant d'effectuer des mesures par échantillonage. L'utilisation de cet outil est une tâche privilégiée du système et elle peut être faite sans rien modifier au système.

GETAB est un appel moniteur qui permet aux utilisateurs d'accéder à certaines tables du système (décrites dans l'annexe B) et par là de collecter des informations sur les performances et l'état du système. Il suffit de fournir le numéro d'identification de la table désirée et GETAB retourne une entrée de cette table.

Ce moyen est assez limité, puisqu'il ne permet l'accès qu'à des données présentes dans des tables du système et de plus, le nombre de tables accessibles par GETAB est assez restreint. Quoi qu'il en soit, il nous paraît intéressant, et nous l'utiliserons pour mesurer certaines valeurs spécifiques du système.

III.1.2.3. L'outil MDDT

Comme l'outil précédent (GETAB), l'outil MDDT est un outil de mesure par échantillonage. Cependant il présente des caractéristiques assez différentes de GETAB.

MDDT (Monitor Dynamic Debugging Technique) peut uniquement être utilisé par des utilisateurs privilégiés du système. Il permet de lire ou d'écrire (attention, dangereux) dans toutes les tables et variables internes quand le système tourne normalement. Il offre l'avantage de permettre l'accès à toutes les tables. Cependant, son principal désavantage est qu'on ne peut l'utiliser que de manière interactive et jamais dans un programme. La collecte de données par cette méthode est donc extrêmement longue et fastidieuse. De plus ce "debugger" présente un danger évident : on peut également écrire dans les tables du système, une fausse manoeuvre peut donc "crasher" le système".

Comme nous disposons d'autres outils beaucoup plus commodes d'utilisation, nous rejetons la possibilité de se servir de MDDT pour le mesureur en lui-même.

Cependant, cet outil peut nous aider pour préciser ou éclaircir certaines informations fournies par les documents de Digital Equipement Corporation. (Par exemple, préciser le contenu de certaines tables du système).

III.1.2.4. L'outil SNOOP

L'outil SNOOP diffère des outils vus jusqu'ici, par le fait qu'il permet la mesure sur événements et non plus par échantillonage.

SNOOP est un appel moniteur qui permet d'ajouter n'importe quelle instruction à un endroit quelconque du code moniteur. Nous appellerons cette instruction une sonde. Introduire une sonde a donc pour effet de modifier le système.

Examinons comment un utilisateur peut se servir de cet outil :

1) Dans le programme de l'utilisateur sont définies différentes routines que celui-ci souhaiterait insérer dans le code moniteur pour effectuer les mesures qu'il désire. Pour cela, il suffit que chaque sonde consiste en un appel à une de ces routines de mesure.

2) Si ces routines de mesures utilisent des tables, des variables ou n'importe quel symbole défini dans le code moniteur, le programme utilisateur doit, grâce au paramètre "snpsy" de l'appel SNOOP, saisir l'adresse correspondant à ce symbole.
   Si ces routines de mesures utilisent des tables ou variables supplémentaires, elles doivent être définies dans le programme utilisateur.

3) Les différentes routines ainsi que les tables et variables supplémentaires dont ces routines se servent doivent être copiées en un endroit de l'espace moniteur défini par le moniteur lui-même (maximum 8 pages).

Ceci est effectué grâce au paramètre ".snplc" de l'appel SNOOP qui signale également à l'utilisateur l'adresse choisie pour le système.

4) Ces routines ayant été au départ relogées par rapport à l'adresse de début du programme utilisateur, il faut, après ce transfert, les reloger par rapport à l'adresse choisie par le moniteur. C'est l'utilisateur lui-même qui doit effectuer cette relocation par rapport à l'adresse qui lui a été fournie par l'appel SNOOP (.snplc).

5) On peut maintenant définir les "break points", c'est-à-dire les endroits dans le code moniteur où les sondes prendront place. Cette définition est réalisée en utilisant l'argument ".snpdb". de l'appel SNOOP.

6) Par le paramètre ".snpib" de l'appel SNOOP, on insère simultanément toutes les sondes aux endroits que l'on vient de définir.

7) Tandis que les routines effectuent leurs mesures, le programme utilisateur s'endort.

8) Quand le temps écoulé pour les mesures est suffisant, on peut enlever les sondes et donc arrêter l'exécution des routines de mesures. Pour faire cela, on utilisera l'appel SNOOP avec comme paramètre ".snprb".

9) On peut maintenant, grâce au paramètre ".snpul", libérer l'espace moniteur qu'occupaient ces routines et tables depuis l'étape 3.

10) Il reste au programme utilisateur à traiter les mesures qui lui ont été fournies et à les imprimer.

Cette méthode pour effectuer des mesures est beaucoup moins restrictive que celles vues jusqu'à présent. Elle permet de lire les tables existantes, d'en définir de nouvelles, de noter des valeurs calculées par le moniteur au moment où elles sont calculées alors qu'elles disparaissent ensuite, de noter le moment où certaines actions sont effectuées (au moyen de DATAI TIM, LOC ou de la variable TODCLK).
Cependant, contrairement à l'appel GETAB, en utilisant SNOOP, il faut écrire les routines soi-même.

Le danger de cette méthode est que l'on modifie le moniteur. Si les routines de mesures sont insérées à des endroits non désirables ou si ces routines inscrivent n'importe quoi dans le code moniteur, cela est généralement fatal pour le bon déroulement de TOPS-20. Il faut bien réfléchir à l'endroit où s'introduisent les sondes. De même, les routines de mesure doivent préserver la valeur de tous les accumulateurs qu'elles utilisent et être très prudentes quant aux modifications des variables et tables du moniteur.

III.2. DESCRIPTION GENERALE DU MESUREUR [DIG1, DIG6]

Nous allons ici décrire de manière générale les mesures que notre mesureur effectue. Pour les personnes initiées et intéressées par la manière dont ce genre de mesures est effectivement implémenté, ce paragraphe constitue une introduction au paragraphe suivant. Pour les autres lecteurs, ce paragraphe de description générale peut être suffisant.

Comme nous venons de l'expliquer, le mesureur est basé sur deux appels moniteurs : GETAB et SNOOP. De plus, pour déterminer le moment d'occurence de certains événements du système, en même temps que l'outil SNOOP, nous utiliserons la variable du système TODCLK. Dans les mesures faites au moyen de SNOOP, nous distinguons :

- la mesure du temps des Entrées/Sorties sur terminal
- la mesure du temps passé dans le "CPU" avant d'en sortir et la mesure des différentes probabilités de transition en sortant du processeur.
- la mesure du temps de service des unités et des canaux.

Dans les mesures faites au moyen de GETAB, nous distinguons :

- la mesure de nombre de prccessus dans chaque file auxiliaire
- la mesure du nombre de processus dans le "balance set".

III.2.1. Mesure du temps des E/S sur terminal

Le temps d'entrée/sortie sur un terminal, est le temps écoulé entre le début de la sortie faite de la part du système sur le terminal et la fin de l'entrée faite par l'utilisateur.

Cependant, la sortie sur terminal n'appelle pas toujours une entrée et de même, une entrée n'est pas toujours appelée par une sortie. Dans ces deux cas, c'est uniquement le temps pris par la sortie ou l'entrée qui sera pris en compte.

Parmis tous les temps que nous avons mesurés, c'est le seul que nous avons repris directement du système. En effet, pour déterminer le mouvement ascendant dans les files (I.2.3.2.), le système doit déterminer le temps pendant lequel le processus s'est bloqué. Si l'on distingue le cas d'une E/S sur terminal, on peut dès lors se servir de ce temps. Nous distinguons pour cette mesure deux valeurs : le nombre de mesures faites, la moyenne. Ces deux valeurs sont calculées séparément pour chaque file auxiliaire (0 à 5).

III.2.2. Mesure du temps passé dans le "CPU" et des différentes probabilités de transition

Nous calculons ici le temps passé par un processus dans le "CPU", avant que celui-ci ne soit alloué à un autre processus. C'est-à-dire le temps pendant lequel un processus occupe de manière contigüe le "CPU".

Ce temps n'étant pas calculé par le système, nous devons le faire nous-même. Nous saisissons l'instant où le processus entre dans le "CPU", l'instant où il en sort et en faisant la différence, nous obtenons le temps désiré.
Nous distinguons ici trois valeurs qui seront calculées séparément, pour chaque file : le nombre de mesures, la moyenne et la variance.

A la sortie du "CPU", nous devons également mesurer les probabilités de transition vers les autres points ou serveurs du modèle. Nous devons mesurer la probabilité qu'a un processus sortant du processeur d'aller :

vers les terminaux (E/S longues)
vers les disques et bandes magnétiques (E/S courtes)
vers le "CPU", c'est-à-dire de retourner dans la file d'attente du "CPU"
vers la file d'attente d'entrée dans le "balance set", (PRE-FIL)

Pour cela, nous comptons le nombre de processus allant dans chaque direction et nous pouvons donc calculer le pourcentage par rapport au nombre total de processus sortant du "CPU". Ces pourcentages sont calculés séparément pour chaque file auxiliaire.

Enfin, quand un processus se dirige vers les disques et bandes, il nous reste à mesurer la probabilité qu'il a d'utiliser telle unité disque (PS : 1, PS : 2, PS : 3, US:, disque montable) ou telle unité bande (MTA0, MTA1). Ce pourcentage est calculé globalement, sans différencier les files. En effet, les probabilités de se diriger vers les terminaux, disques et bandes ... peuvent fortement différer selon la file du processus, car ces files reflètent le degré d'interactivité du processus. Par contre, la probabilité d'utiliser telle ou telle unité devrait être sensiblement uniforme, quelle que soit la file auxiliaire.

III.2.3. Mesure du temps de service des unités et des canaux

Pour les unités et les canaux, nous devons mesurer le temps de positionnement de l'unité et le temps de transfert d'une page. Comme pour la mesure du temps passé dans le "CPU", on procède en saisissant l'instant de début et l'instant de fin et en faisant la différence entre ces deux temps. Pour ces mesures, nous calculons les trois valeurs précédentes, à savoir, le nombre de mesures, la moyenne et la variance.

Pour les unités disques, le positionnement correspond au déplacement du bras pour trouver le cylindre correct. Mais qu'est en réalité le positionnement d'une bande ? Ce n'est pas uniquement le défilement de la bande jusqu'à l'endroit demandé. Pour positionner une bande, il faut lire la bande pour voir où on se trouve.

Pour une bande, le système convertit donc la demande de positionnement en un ensemble de demandes de transferts. Puisque le positionnement d'une bande n'existe que logiquement (pour l'utilisateur) et pas physiquement (pour le dérouleur de bande), nous ne devons pas le mesurer. Tout "positionnement" occupe en plus de l'unité bande, le canal. Les deux dérouleurs de bandes étant connectés sur le même canal, il n'est donc plus nécessaire de différencier MTA0 et MTA1 (dans les probabilités de transition, III.2.2. et pour la suite). Nous effectuons les mesures du temps de positionnement pour les unités disques (PS : 1, PS : 2, PS : 3, US :, disque montable) et du temps de transfert pour les canaux (0, 1, 2, 3).

Cependant, l'algorithme d'optimisation du positionnement du bras des disques (SCAN algorithme expliqué en I.5.2.) nous fait ajouter une autre mesure aux deux que nous venons de décrire : le nombre moyen de demandes transférées de la PWQ vers la TWQ d'une unité, lors du positionnement de celle-ci sur un cylindre. Cette mesure est effectuée séparément pour chaque unité disque et dépend fortement de la charge. Nous assimilons, pour notre facilité, à ces demandes transférées de la PWQ vers la TWQ les demandes pour lesquelles un positionnement n'est pas nécessaire. En réalité, de telles demandes transitent directement vers la TWQ sans passer par la PWQ. Cependant, comme nous travaillons en moyenne, nous pouvons nous permettre de les mettre dans la PWQ et lors d'un positionnement, de transférer une demande de plus de la PWQ vers la TWQ. Cela nous permet de pouvoir enregistrer toutes les demandes d'entrée/sortie directement dans la PWQ.

Rappel : pour nous, la PWQ est la file devant chaque disque et la TWQ est la file devant chaque canal.

De plus, lorsque nous étudions de très près le système, nous nous rendons compte, qu'il existe des cas où l'on ajoute des demandes supplémentaires dans la TWQ d'une unité.

Ces cas peuvent se produire quand on vérifie le "Home block" d'une unité, ou quand le système élimine complètement une demande (IORB) qui a été servie. D'autre part, il se peut que l'opérateur décide de supprimer une unité. Dès lors, le système doit annuler toutes les demandes qui ont été produites pour ces unités. Bien que nous estimons que ces trois dernières situations ne se produisent que dans de rares cas, nous les mesurerons distinctement pour chaque unité. Nous déciderons d'après les résultats obtenus par ces mesures, si nous devons en tenir compte ou non dans notre modèle.

III.2.4. Mesure du nombre de processus dans le "balance set"

Un appel moniteur GETAB nous fournit l'intégration (sur le temps écoulé depuis la génération du système) du nombre de processus dans le "balance set". Nous n'avons donc pas besoin de calculer cette valeur nous-mêmes (par SNOOP). Il suffit d'effectuer un premier appel, puis après un certain temps d'en effectuer un second, de soustraire le premier résultat fourni par GETAB du second, et enfin de diviser par le temps écoulé entre les deux appels. On obtient ainsi la moyenne du nombre de processus dans le "balance set".

III.2.5. Mesure du nombre de processus dans chaque file auxiliaire

La mesure que nous voudrions obtenir est le nombre de processus dans chaque file appartenant à la GO-LISTE et aux listes d'attente d'E/S sur terminal.

Cette mesure est assez complexe et risque de prendre beaucoup de temps "CPU" et de perturber le système. De plus, le temps nous étant imparti ne nous permettant pas d'élaborer cette mesure supplémentaire, nous avons choisi de la simplifier.

En effet, GETAB est capable de nous fournir deux indications sur ce nombre de processus :
le "load-average", différencié uniquement en "high-queues" et "low-queues", les "high-queues" étant les quatres premières files auxiliaires (0 - 1 - 2 - 3, les plus interactives) et les "low-queues" les deux dernières (4 - 5). Rappelons également que le "load-average" est une estimation du nombre de processus dans la GO-LISTE (cfr. I.2.3.2.). Il nous est fourni par GETAB pour un instant déterminé. Nous devons donc régulièrement mesurer ces valeurs, les cumuler et terminer en divisant ces valeurs cumulées par le nombre de mesures prises (mesure par échantillonage).

Une autre valeur que pous pouvons estimer, est le nombre de processus aux terminaux. En effet, la théorie des processus stochastiques nous apprend que le nombre moyen de processus ($\bar{n}$) dans une station quelconque (station : serveur et file d'attente) est égal au taux d'arrivée des processus dans cette station (λ), multiplié par le temps moyen passé dans cette station ($\bar{u}$) c'est-à-dire :

$$(1) \quad \bar{n} = \lambda \bar{u}$$

Nous savons aussi, que le temps moyen passé dans une station est égal à la somme du temps moyen d'attente ($\bar{w}$) et du temps moyen de service de la station ($\bar{s}$)

$$(2) \quad \bar{u} = \bar{w} + \bar{s}$$

La station "TTY" étant une station avec un nombre de serveurs infini, il n'y a pas de temps d'attente.
Donc :

$$(3) \quad \bar{w} = 0$$

Et l'on obtient

$$(4) \quad \bar{n} = \lambda \bar{s}$$

Il ne nous reste plus qu'à définir à quoi correspondent λ et $\bar{u}$ pour la station "TTY"

- $\bar{u}$ est le temps moyen d'une E/S sur terminal.
  Or, nous avons déjà mesuré cette valeur par file auxiliaire (III.2.1.). Nous possédons donc $\bar{u}_i$ avec i = 0 ... 5 = n° de la file auxiliaire.

- λ, le taux d'arrivée des processus aux terminaux est égal au nombre d'E/S effectuées sur les terminaux, divisé par le temps total pendant lequel nous avons effectué nos mesures. Or nous avons déjà calculé le nombre d'E/S effectuées par file, lors de la mesure du temps moyen d'E/S sur terminal (III.2.1.). Nous possédons donc aussi λ_i pour i = 0 ... 5 = numéro de la file auxiliaire.

Nous sommes donc capable d'obtenir les "high-queue" et "low-queue load-average" et de calculer le nombre de processus aux terminaux par files mais aussi globalement (il suffit de cumuler sur "i" les " i" et "$\bar{u}i$"). C'est tout ce qu'il nous faut puisque notre modèle ne s'occupe que des processus dans la GO-LISTE et dans les files d'attente pour E/S sur terminal. Cependant nous savons que ce n'est qu'une estimation du nombre moyen de processus et de plus, nous ne pouvons déterminer exactement le nombre de processus dans la GO-LISTE pour chaque file auxiliaire. Cela n'est fait qu'en différenciant "high-queues" et "low-queues". Nous verrons plus tard (lors de l'application de ces paramètres à notre modèle) ce que nous pourrons en faire.

III.2.6. Mesure du pourcentage d'utilisation des serveurs

Pour vérifier que le modèle que nous allons élaborer (chap. IV) est correct par rapport au système réel, nous calculons le pourcentage d'utilisation des "serveurs" terminaux, "CPU", unités canaux. Pour les terminaux, cette mesure n'est pas évidente. En effet, elle devrait être effectuée en différenciant chaque terminal, or nous n'avons aucun moyen de faire cette différenciation. Une autre façon de faire serait d'effectuer la mesure globalement pour tous les terminaux et d'en tirer une moyenne en divisant par le nombre de terminaux actifs, or nous ne disposons pas de ce nombre, il faudrait le calculer.

Comme cette mesure alourdirait considérablement notre mesureur, et que nous jugeons qu'il est suffisant de disposer du pourcentage d'utilisation du "CPU", des unités et des canaux, nous avons décidé de ne pas faire cette mesure.

Pour les trois autres serveurs, la mesure est simple. Il suffit pour une période donnée de mesurer le temps pendant lequel le "CPU" est occupé, le temps pendant lequel chaque unité est occupée (temps de positionnement + temps de transfert) et le temps pendant lequel chaque canal est occupé (temps de transfert) et de diviser ces trois valeurs par la période de temps pendant laquelle on effectue les mesures.

III.3. DESCRIPTION DES MECANISMES DU MESUREUR [DIG1, DIG6]

Nous venons de décrire les mesures que nous effectuons. Expliquons maintenant les mécanismes qui nous permettent d'implémenter ces mesures. Rappelons que cette partie s'adresse principalement aux lecteurs intéressés par l'implémentation d'un tel mesureur. Nous aurions pu ne pas nous attarder sur de telles choses, cependant, l'élaboration du mesureur ayant été notre plus gros travail et notre plus gros soucis, nous avons jugé utile d'y consacrer quelques pages.

III.3.1. Structure du mesureur

La structure de ce programme de mesure a été déterminée par la manière dont on a utilisé l'outil SNOOP. Quant à l'outil GETAB, il est inséré dans la partie utilisateur du programme, qui est la dernière partie.

Partie I

Cette partie comprend en 1er lieu, la définition des données utilisées par les routines d'appel à SNOOP. Ensuite viennent les routines effectuant les appels à SNOOP déjà décrits précédemment (I.1.2.4.). C'est-à-dire

1) insérer les pages dans le moniteur
2) définir les endroits des "Break points"
3) reloger les routines de mesure
4) acquérir les adresses de certaines variables du moniteur
5) effectuer l'insertion des routines de mesure
6) retirer les sondes
7) libérer l'espace moniteur.

Partie II

Dans cette partie se trouvent les pages à insérer dans le moniteur. Elle comprend en fait trois paragraphes :

1) Déclaration des données utilisées par les routines de mesures (variables, tableaux, constantes)
2) Les différentes sondes qui sont insérées dans le code moniteur et qui consistent en fait en des appels aux routines de mesures qui les suivent
3) Les routines de mesures exécutées grâce aux appels SNOOP (Ces mesures sont décrites plus loin).

Partie III

C'est la partie utilisateur du programme. Cette partie gère les routines d'appels à SNOOP (Partie I) et les résultats fournis par les mesures (Partie II).

De plus, c'est ici que sont effectués les appels à GETAB. Cette partie comprend elle-même trois paragraphes

- définition des données utilisées par le programme principal
- routine d'erreur lors des appels moniteur à SNOOP.
- le programme principal :

1) Initialisation des variables utilisées par les routines de mesures.
2) Appels aux routines de la partie I qui elles-mêmes feront démarrer les routines de la partie II (Points 1 à 5 de la Partie I).
3) Premiers appels à GETAB pour saisir les informations données sous forme d'intégrale (mesure du nombre de processus dans le "balance set" et du nombre de processus dans H.Q et L-Q).
4) Nous couplons ici les appels qui doivent se faire périodiquement par GETAB (mesure du "load-average") et la période de temps pendant laquelle le programme doit s'endormir pour permettre aux routines de mesures de s'effectuer.

5) Deuxième appel à GETAB pour saisir les informations données sous forme d'intégrale.

6) Appels aux deux dernières routines de la partie I pour arrêter les mesures et libérer l'espace moniteur.

7) Calcul et impression des résultats.

III.3.2. Les temps de service("TTY", "CPU", unités et canaux)

Les temps de service sont mesurés au moyen de SNOOP. Pour déterminer le temps des E/S sur terminal, il nous suffit d'une routine de mesure qui saisit ce temps au moment où il est calculé par le système (cfr. III.2.1.). Pour les autres serveurs ("CPU", unités et canaux), il nous faut deux routines, la première saisissant l'instant où débute le service, la seconde l'instant où se termine le service.

Pour ces temps de service, nous devons calculer 3 valeurs : nombre de mesures, moyenne et variance. Et cela, en les différenciant soit par file ("TTY, "CPU") soit par unité (positionnement de l'unité et transfert de l'unité et du canal). Nous utilisons pour cela, trois variables multiples (tables) : la première est incrémentée à chaque mesure, dans la deuxième, on cumule le temps mesuré, dans la troisième, on cumule le temps mesuré élevé au carré. Ces incrémentations et cumuls sont effectués dans la routine de fin de service ou évidemment dans l'unique routine de mesure de temps de service. Quand le temps alloué aux mesures est terminé, la partie utilisateur du programme peut alors diviser les valeurs cumulées par le nombre de mesures et obtenir ainsi moyennes et variances.

III.3.3. Sortie du "CPU" et probabilités de transitions

La sortie du "CPU" est un peu plus particulière que ce que nous venons de décrire. En effet, nous la mesurons à plusieurs endroits dans le système selon la direction vers laquelle va se diriger le processus.:

- terminaux
- disques et bandes magnétiques
- "CPU"
- file d'attente d'entrée dans le "balance-set". (PRE-FIL)

Nous avons donc quatre routines pour mesurer la sortie du "CPU". Dans ces routines, nous mesurons également les probabilités de transition . Nous associons un compteur propre à chaque sortie en plus du compteur général du nombre de mesures du temps de service du "CPU". Ces compteurs sont incrémentés à chaque passage dans la routine. Quand les mesures sont terminées, il suffit à la partie utilisateur du programme (partie III) de faire pour chacun de ces compteurs propres, le rapport sur le nombre total de processus sortant du "CPU" (compteur de nombre de mesures du temps de service), afin d'obtenir ainsi les différentes probabilités de transitions.

III.3.4. Détermination du numéro de file auxiliaire

Nous devons pour différencier les mesures du temps d'E/S sur terminal et du temps de service "CPU", disposer du numéro de la file auxiliaire à laquelle appartient le processus mesuré. La table FKQ2 (cfr. annexe A) contient ce numéro pour chaque processus. Il nous suffit d'indexer cette table par le numéro du processus pour l'obtenir. Or, si quand on mesure le temps de service des "TTYs", on dispose de ce numéro de processus, par contre, pour les différentes mesures de la fin du temps "CPU", ce n'est pas toujours le cas. Nous devons donc le déterminer, au début de la mesure (entrée dans le "CPU") et le mémoriser dans une variable (variable des pages à insérer dans le moniteur). A la sortie du "CPU", nous disposerons toujours de cette variable, elle n'aura pu avoir été modifiée.

III.3.5. Détermination de l'unité

Nous devons différencier les mesures portant sur les probabilités de transition vers les disques et bandes et les mesures portant sur le positionnement des unités et le transfert par rapport à l'unité à laquelle elles se rapportent (sauf pour les bandes cfr. 2.3.). Il faut donc que nous puissions déterminer sur quelle unité la demande porte ; c'est-à-dire PS:1, PS:2, PS:3, US, le disque montable pour les disques (PS et US étant la structure), ou les bandes.

Pour cela, nous attribuons un numéro à chaque unité (respectivement dans l'ordre ci-dessus : 0, 1, 2, 3, 4, 5). Grâce à ce numéro nous pouvons indexer des tables.

Premièrement, nous pouvons différencier disques et bandes grâce au "status" du bloc de données des unités ("UDB" cfr. annexe A). Ensuite, nous traitons séparément le cas des disques et le cas des bandes.

- Pour les disques nous procédons de la manière suivante : Lorsque le système effectue la mise en file d'une demande d'entrée-sortie courte, il doit déterminer l'unité à laquelle elle se rapporte, il détermine l'adresse du "UDB" de l'unité. Dans cet "UDB" se trouve un pointeur vers le bloc décrivant sa structure ("SDB"). Cependant, ce pointeur n'est pas direct, il contient dans sa première moitié, le numéro de l'unité à l'intérieur de la structure et dans la deuxième moitié, le numéro de la structure. Pour obtenir l'adresse du "SDB", nous devons indexer la table "STRTAB" par ce numéro de structure. Nous trouvons enfin à l'intérieur du "SDB", le nom de la structure (PS ou US). Le disque montable s'appliquera à toute unité n'étant ni la première, la deuxième ou la troisième unité de PS, ni la première unité de US.
- Quant aux bandes magnétiques, nous avons vu précédemment (III.2.3.) qu'il ne faut pas les différencier.

Cependant, nous nous rendons compte que cette identification de l'unité est assez lourde pour le mesureur. De plus, elle doit être effectuée pour plusieurs mesures. Nous devons nous limiter à la calculer une seule fois et la mémoriser dans un endroit toujours accessible par notre mesureur : nous déterminons l'unité à la sortie du "CPU" vers les unités (mesurée lors de la mise en file de la demande) et nous mémorisons le numéro lui étant associé, dans une partie du UDB non utilisée.

III.3.6. Pourcentage d'utilisation des serveurs ("CPU", unités et canaux)
Pour calculer ce pourcentage, nous devons mesurer le temps total pendant lequel les serveurs "CPU", unités et canaux sont utilisés. Or lors de la mesure du temps de service de ces serveurs, nous cumulons chaque fois qu'un processus utilise le serveur, le temps pendant lequel il l'occupe. Il suffit donc que la partie utilisateur du programme (partie III) capte ces valeurs cumulées et les divise par le temps total des mesures. Nous ne devons donc pas ajouter de routines pour ces mesures.

III.4. RESULTATS PRODUITS PAR LE MESUREUR

Nous avons exécuté notre mesureur à différents moments de la journée et de la semaine. A chaque fois, il effectuait des mesures durant une heure. Malheureusement, vu les difficultés que nous avons rencontrées et donc le temps qu'il nous a fallu pour mettre au point ce mesureur, nous sommes arrivés dans une période creuse pour l'utilisateur du DEC SYSTEM-20 : les vacances. Seuls, quelques rares mémorants travaillaient aux terminaux. Vu la charge peu élevée, nous n'avons pu obtenir de résultats très enrichissants : ce qui nous intéressait surtout, c'était de mesurer un système en pleine activité, et même surchargé. De plus, nous désirions examiner toutes les mesures par rapport à la charge du système, examiner si certaines en dépendaient et d'autres pas. Nous aurions voulu pour les paramètres variant avec la charge, établir des courbes de variation par rapport à cette charge. Malheureusement, tout cela a été rendu impossible à réaliser.

Nous nous contenterons donc ici, de donner une tendance générale des résultats que nous avons obtenus, en espérant qu'en les extrapolant avec une certaine prudence, ils pourront nous donner des indications sur le système chargé, de manière normale. Signalons que tous ces résultats se trouvent de manière intégrale en annexe.

III.4.1. Les nombres de processus

Il s'agit des "loads averages" dans les "high-queues" et "low-queues", du nombre de processus aux terminaux (par file ou pas) et du nombre de processus dans le "balance set". Vu les circonstances, c'est la mesure la moins parlante. En effet, en plus que de dépendre immédiatement de la charge du système, elle en est un indicateur évident. Nous allons donner pour chacune de ces mesures, une fourchette des résultats que nous avons obtenus.

Nous avons effectué 6 fois les mesures pendant une heure (3 fois le matin et 3 fois l'après-midi). Les fourchettes ont été établies en prenant parmi ces 6 mesures la valeur minimale et la valeur maximale du nombre mesuré.
(Rappelons que H-Q sont les files auxiliaires 0-1-2 et 3 et L-Q les files auxiliaires 4 et 5 (cfr. I.2.3.).

Commençons par les nombres différenciés par files auxiliaires.

. Load-average (nombre de processus dans la GO-LISTE).

High-queues : [1,39 ; 3,36]
Low-queues : [1,93 ; 2,82]

. Nombre de processus aux terminaux

file 0 : [0,00 ; 0,00]
file 1 : [15,00 ; 20, 58]
file 2 : [3,61 ; 6,70] 23,14
file 3 : [0,52 ; 2,07]
file 4 : [0,15 ; 0,31]
file 5 : [0,00 ; 0,03]

Nous constatons que, pour les valeurs les plus basses de la charge, c'est dans les "low-queues" qu'il y a le plus de processus dans la GO-LISTE, tandis que pour les valeurs plus élevées (qui restent quand même très basses), la situation se renverse. Cela est dû au fait que pour faire "tourner" le système, il doit toujours y avoir un minimum de processus. Ces processus n'étant jamais interactifs, ils se situent dans les "low-queues". Donc, comme les étudiants exécutent essentiellement des processus interactifs, plus il y a d'étudiants, plus le nombre de processus dans le système augmente, et cela en faveur des H-Q,. Si nous avions pu effectuer des mesures sur charge élevée, nous aurions dû pouvoir confirmer cette tendance.

Quant au nombre de processus aux terminaux, il est toujours très faible si pas nul pour les files auxiliaires spéciales 0 et 5, tandis qu'il est très élevé pour la file 1 (on s'y attendait). Pour la file 2, il y a une valeur très spéciale qui n'est pas reprise dans la fourchette. En effet, pour une des mesures, on obtient 23,14 processus dans cette file, ce qui est même supérieur au nombre de processus dans la file 1. Ce cas ne s'est présenté qu'une seule fois.

Examinons maintenant le nombre total de processus dans le système, sans différencier les files auxiliaires.

- Nombre de processus aux terminaux.

$$[19.45 \ ; \ 39.95]$$

- Load-average H-Q et L-Q : (nombre de processus dans la GO-LISTE)

$$[3,83 \ ; \ 6,18]$$

- Nombre moyen de processus dans le "balance set" :

$$[3,84 \ ; \ 6,07]$$

Nous remarquons que la plus grande partie des processus se trouvent dans les listes d'attente pour entrées (sorties sur terminal) (cfr. I.2.2.). En effet, le temps passé pour faire ces E/S est assez élevé par rapport aux autres temps de service des processus appartenant à la GO-LISTE ("CPU", disques, canaux). Cette différence devrait encore s'accroître avec la charge.

On peut aussi constater que pratiquement tous les processus de la GO-LISTE se trouvent dans le "balance set". La charge étant très basse, tous les processus de la GO-LISTE peuvent entrer simultanément en mémoire centrale, ils n'ont pas à effectuer d'attente pour cette raison. Nous sommes donc dans l'incapacité de tirer quelle conclusion que ce soit à ce sujet.

III.4.2. Temps passé au terminal et dans le "CPU"

Comme nous le savons, pour chaque file auxiliaire, nous avons mesuré deux valeurs : le nombre de fois que le mesureur a effectué la mesure (c'est-à-dire le nombre de fois qu'un processus entre dans le serveur) et la moyenne du temps passé (+ la variance pour le temps "CPU"). Nous allons donc présenter dans deux tableaux ces trois valeurs, mais cumulées pour toutes les exécutions de notre mesureur.

* Temps passé au terminal

	Nombre de mesures	moyenne (ms)
File 0	0	0
File 1	215 891	1780
File 2	31 226	3184
File 3	12 595	1965
File 4	280	18004
File 5	1 067	150

Nous constatons, que les processus de la file 0 ne font jamais d'interaction avec les terminaux. Plus un processus est "compute-bound", c'est-à-dire plus le numéro de la file est élevé, moins il y a d'E/S au terminal, ce qui était attendu. Pourtant, la file 5 est un peu particulière. En effet, pour certaines exécutions du mesureur, nous avons un nombre très bas de mesures (1 ou 20). Pour d'autres, un nombre très élevé (700). Pour tirer des conclusions significatives à propos de cette file, il aurait été indispensable de faire des mesures en charge élevée.

Quant à la moyenne du temps passé au terminal, elle est très longue pour les processus de la file 4 (les plus "compute-bound") et très courte pour ceux de la file 5. Pour les processus interactifs, c'est dans la file 2 qu'on passe le plus de temps au terminal. Ces conclusions se dégagent globalement mais aussi pour chaque exécution d'une heure. Cependant nous ne pouvons rien déduire de plus.

* Temps passé dans le "CPU"

	Nombre de mesures	moyenne (ms)	variance (ms*ms)	σ/μ : coefficient de variation
File 0	18 119	6.9	77 83	1,278
File 1	351 504	7.9	132 00	1,454
File 2	124 965	7.5	164,50	1,71
File 3	178 351	10	236,16	1,536
File 4	167 202	12.7	312.33	1,391
File 5	88 718	14.9	391.33	1,327

Signalons que nous avons rajouté une colonne ; celle-ci comprend les coefficients de variation (σ/μ) calculés directement à partir des colonnes 2 et 3.

Hormis la file 0, où il n'y a pas beaucoup de processus, on peut dire que le nombre de mesures diminue quand le numéro de la file augmente. Mais peut-on dire pour autant que moins la file est interactive, moins elle contient de processus ? Non, cela dépend aussi de la fréquence de passage des processus dans le "CPU". Or, on remarque que plus un processus est "compute-bound", plus il passe de temps dans le "CPU". On ne peut donc pas tirer de conclusions, quant au nombre de processus dans chaque file, on peut juste remarquer que les files 0 et 5 sont moins peuplées que les autres (surtout la 0) et que la file 1 (la plus interactive) est très peuplée.

Nous pensons que le contre-temps occurencé par la faiblesse de la charge n'a eu que très peu d'influence sur les temps moyens des mesures. Le seule chose, c'est que cet échantillon formé par les utilisateurs du système n'est peut être pas très représentatif d'une charge normale. Cependant pour la raison que nous avons déjà expliquée en III.4.1., la répartition des processus entre les différentes files ne doit pas du tout être la même dans le système "tournant" habituellement.

Nous avons essayé de déterminer par test statistique si les moyennes obtenues par file étaient significativement différentes. Pour cela, nous avons utilisé les méthodes d'estimations utilisant les intervalles de confiance et s'appliquant à des échantillons de populatinn nombreux, à grand effectif. Nous avons estimé la différence des moyennes de la file 1 et de la file 0 (1) ainsi que de la file 1 et la file 2 (2).

Si l'intervalle d'estimation de la différence ne couvre pas la valeur zéro, c'est que les 2 moyennes sont significativement différentes au niveau d'erreur choisi.

Voici la formule à appliquer (il s'agit d'un cas particulier de l'estimation d'une combinaison linéaire des moyennes de p populations).

$$\Pr\left[m_1 - m_0 - Q_G\,(1 - 0{,}025)\sqrt{\frac{S1^2}{n_1} + \frac{S0^2}{n_0}} \leqslant \mu_1 - \mu_0 \leqslant m_1 - m_0 + Q_G\,(1 - 0{,}025)\sqrt{\frac{S1^2}{n_1} + \frac{S0^2}{n_0}}\right] = 0{,}95$$

(1)
$$\Pr\left[7{,}9 - 6{,}9 - 1{,}96 \times \sqrt{\frac{132}{351504} + \frac{77{,}83}{18119}} \leqslant \mu_1 - \mu_0 \leqslant 7{,}9 - 6{,}9 + 1{,}96 \times \sqrt{\frac{132}{351504} + \frac{77{,}83}{18119}}\right] = 0{,}95$$

$$\Pr\left[1 - 0{,}13395593 \leqslant \mu_1 - \mu_0 \leqslant 1 + 0{,}13395593\right] = 0{,}95$$

$$\Pr\left[0{,}86604407 \leqslant \mu_1 - \mu_0 \leqslant 1{,}13395593\right] = 0{,}95$$

On remarque donc que dans 95 % des cas, l'intervalle sera compris entre 0,866 et 1,133, et ne couvrira donc pas la valeur 0. On peut donc dire que les moyennes μ_0 et μ_1 des files 0 et 1 sont significativement différentes.

Il en est de même pour les moyennes μ_1 et μ_2 des files 1 et 2.

(2) $$\Pr\left[7,9 - 7,5 - 1,96 \times \sqrt{0,0012192703} \leq \mu_1 - \mu_2 \leq 7,9 - 7,5 + 1,96 \times \sqrt{0,0012192703}\right] = 0,95$$

$$\Pr\left[0,4 - 0,06843937 \leq \mu_1 - \mu_2 \leq 0,4 + 0,06843937\right] = 0,95$$

$$\Pr\left[0,33156063 \leq \mu_1 - \mu_2 \leq 0,46843937\right] = 0,95$$

L'intervalle ne recouvre pas 0.

On pourrait continuer pour les moyennes des autres files 2 à 2, on obtiendrait la même conclusion : les moyennes sont significativement différentes. Ceci est principalement dû au fait que les échantillons sont très nombreux, ce qui rend les intervalles de confiance très petits.

Nous avons aussi jugé utile de calculer un intervalle de confiance pour l'estimation du coefficient de variation. Il faut signaler que cette méthode s'applique à une population sensiblement normale. Le coefficient de variation correspond à $\frac{\sigma}{\mu}$ (écart-type sur moyenne).

Voici la formule :

$$\Pr\left[\frac{S}{m} - Q_G\,(1-\alpha'')\,\frac{S/m}{\sqrt{2n}} \leq \frac{\sigma}{\mu} \leq \frac{S}{m} + Q_G\,(1-\alpha')\,\frac{S/m}{\sqrt{2n}}\right] = 1 - \alpha' - \alpha''$$

Ce qui donne pour la file 0 (en prenant $\alpha' = \alpha'' = 0,025$) :

$$\Pr\left[\frac{\sqrt{77,83}}{6,9} - 1,96 \times \left(\frac{\sqrt{77,83}/6,9}{\sqrt{2 \times 18119}}\right) \leq \frac{\sigma}{\mu} \leq \frac{\sqrt{77,83}}{6,9} + 1,96 \times \left(\frac{\sqrt{77,83}/6,9}{\sqrt{2 \times 18119}}\right)\right] = 0,95$$

$$\Pr\left[1{,}2785697 - 0{,}01316432 \leq \frac{\sigma}{\mu} \leq 1{,}2785697 + 0{,}01316432\right] = 0{,}95$$

$$\Pr\left[1{,}2654054 \leq \frac{\sigma}{\mu} \leq 1{,}291734\right] = 0{,}95$$

De même, voici l'intervalle de confiance pour l'estimation du coefficient de variation du temps "CPU" de la file 1 :

$$\Pr\left[\frac{\sqrt{132}}{7{,}9} - 1{,}96 \times \frac{\sqrt{132}/7{,}9}{\sqrt{2 \times 351504}} \leq \frac{\sigma}{\mu} \leq \frac{\sqrt{132}}{7{,}9} + 1{,}96 \times \frac{\sqrt{132}/7{,}9}{\sqrt{2 \times 351504}}\right] = 0{,}95$$

$$\Pr\left[1{,}4543197 - 0{,}00339966 \leq \frac{\sigma}{\mu} \leq 1{,}4543197 + 0{,}00339966\right] = 0{,}95$$

$$\Pr\left[1{,}45092 \leq \frac{\sigma}{\mu} \leq 1{,}4577194\right] = 0{,}95$$

L'intervalle est ici plus précis vu que la table de l'échantillon est beaucoup plus grande (351504 par rapport à 18119 pour la file 0).

On doit dès lors rejeter l'hypothèse que le coefficient de variation du temps de service "CPU" est égal à 1 et donc que sa distribution est exponentielle. Cependant, le coefficient étant compris entre 1 et 2, on peut espérer que l'approximation exponentielle ne soit pas trop mauvaise.

III.4.3. Probabilités de transition à la sortie du "CPU"

A la sortie du "CPU", nous avons mesuré les probabilités de transition en les différenciant par files. Cependant, pour les transitions vers les différentes unités, nous n'avons pas jugé utile de faire cette répartition (cfr. III.2.2.). Nous allons ici donner des moyennes des pourcentages de processus allant dans chaque direction.

"CPU" →	Terminaux	"CPU"	file d'allocation mémoire	unités
file 0	0	0.65	0.01	99.34
file 1	61.61	0.62	0.02	37.75
file 2	14.89	0.61	0.08	84.42
file 3	7.00	0.30	0.16	92.54
file 4	0.15	0.64	0.14	99.01
file 5	1.20	0.45	0.35	98.00

A part pour les files spéciales (0 et 5), nous constatons que le pourcentage de processus allant vers les unités augmente avec le numéro de la file et que par contre, le pourcentage allant vers les terminaux diminue. Cela, on s'y attendait : la file 1 est la plus interactive et la file 4 la plus "compute-bound". Les processus de la file 0 n'ont pas d'interaction avec les terminaux (ce qui confirme la même conclusion que dans III.4.2.) Quant à la file 5, nous avons fait une moyenne, mais cela n'est pas très représentatif car dans les différentes exécutions de notre mesureur, les pourcentages pour cette file varient beaucoup. Nous remarquons également que les cas où un processus épuise son "quantum" que ce soit son "quantum" de "CPU" ou son "balance set Hold Quantum" arrivent très rarement (transitions "CPU" et allocation mémoire).

Examinons maintenant les pourcentages de processus allant vers chaque unité.

vers PS:1 : 32,13

vers PS:2 : 27,23

vers PS:3 : 27,83

vers US : 7,76

vers le disque montable : 0,02

vers les bandes : 5,03

Ce sont les disques avec la structure PS qui sont les plus utilisés. Pour les bandes, le pourcentage varie très fort parmi les différentes exécutions. En effet, il suffit que l'opérateur lance un sauvetage ou "back-up" pour que le pourcentage de processus allant vers les bandes grimpe en flèche. Pour illustrer ce phénomène, précisons la fourchette obtenue par les exécutions du mesureur : $[0,85 ; 17,90]$. Par contre, les autres valeurs varient très peu.

Pour ces mesures, nous estimons que la charge n'a aucune influence. La seule chose qui peut poser problème, c'est que l'échantillon formé par les utilisateurs travaillant sur le système pendant les vacances n'est peut être pas significatif.

III.4.4. Mesures ayant trait aux unités et aux canaux

Nous avons mesuré pour chaque unité disque son temps de positionnement, ainsi que le nombre moyen de processus qui passaient de la PWQ vers la TWQ lors d'un positionnement. Et pour chaque canal le temps de transfert. A ces mesures s'en ajoutaient trois autres. En effet, en étudiant le système, nous nous sommes aperçus que parfois, il ajoutait des demandes dans la TWQ (deux cas) et que parfois, il supprimait toutes les demandes pour une unité (cfr. III.2.3.).

Après avoir exécuté notre mesureur, nous constatons que ces cas ne se produisent jamais, ou du moins, on ne les a jamais constatés dans nos mesures.

Examinons maintenant les temps de positionnement et de transfert. De la même manière que pour le temps passé dans le "CPU" (III.4.2.), nous allons présenter dans deux tableaux (positionnement et transfert) les trois valeurs mesurées, c'est-à-dire le nombre total de mesures, la moyenne et la variance, plus le coefficient de variation calculé directement (σ/μ).

* Temps de positionnement

	nombre de mesures	moyenne ms	variance ms * ms	coefficient de variation
PS1	107 076	16.15	157.16	0,776
PS2	90 096	16.50	158.16	0,762
PS3	85 751	17.41	169.16	0,747
US	18 417	20.46	177.50	0,651
MONT	68	10.10	48.00	0,685

Nous considérons que le temps de positionnement du bras d'un disque devrait diminuer avec l'augmentation de la charge. Plus la charge est élevée, plus il y aura de demandes d'entrée/sortie. Ces demandes se distribuant sur les cylindres du disque, elles auront alors plus de chances de se rapprocher l'une de l'autre. Donc, le temps de positionnement s'en trouvera réduit.

Ce phénomène se constate d'ailleurs aussi parmi les différentes unités : plus une unité est sollicitée, moins elle prendra de temps pour se positionner (la plus demandée est l'unité PS1, elle a aussi le déplacement minimum).

Pourtant, c'est le disque montable qui est le moins utilisé, et c'est aussi lui qui déplace le plus rapidement son bras. Vu le peu de mesures qui ont été relevées sur cette unité, nous estimons que nous ne pouvons tirer de conclusions valables.

Nous avons calculé un intervalle de confiance au niveau d'erreur de 5 % pour l'estimation du coefficient de variation du temps de positionnement pour PS:1.

$$\Pr\left[\frac{S}{m} - Q_G\,(1 - 0{,}025)\,\frac{S/m}{\sqrt{2n}} \leqslant \frac{\sigma}{\mu} \leqslant \frac{S}{m} + Q_G\,(1 - 0{,}025)\frac{S/m}{\sqrt{2n}}\right] = 0{,}95$$

$$\Pr\left[0{,}776 - 1{,}96 \times \left(\frac{0{,}776}{\sqrt{2 \times 107076}}\right) \leqslant \frac{\sigma}{\mu} \leqslant 0{,}776 + 1{,}96 \times \left(\frac{0{,}776}{\sqrt{2 \times 107076}}\right)\right] = 0{,}95$$

$$\Pr\left[0{,}776 - 0{,}00328667 \leqslant \frac{\sigma}{\mu} \leqslant 0{,}776 + 0{,}00328667\right] = 0{,}95$$

$$\Pr\left[0{,}77271333 \leqslant \frac{\sigma}{\mu} \leqslant 0{,}77928667\right] = 0{,}95$$

Nous devons donc rejeter l'hypothèse que le coefficient de variation du temps de positionnement de PS:1 est égal à 1 et donc que sa distribution est exponentielle. Cependant, le coefficient étant compris entre 0 et 1, on peut espérer que l'approximation exponentielle ne soit pas trop mauvaise.

* Temps de transfert

	nombre de mesures	moyenne	variance	coefficient de variation
CANAL 0	292 407	16.17	517.83	1,407
CANAL 1	198 216	17.71	707.00	1,496
CANAL 2	201 269	17.40	626.00	1,437
CANAL 3	39 202	71.45	6507.16	1,128

Nous constatons que le temps de positionnement et le temps de transfert sont à peu de choses près égaux. Pour les bandes, le transfert est plus long, cela est normal. La charge ne doit pas avoir de conséquences sur ce temps, sauf si l'échantillon n'est pas représentatif de la charge habituelle.

Nous pouvons constater que les coefficients de variation sont compris entre 1 et 2. Quoique rien ne l'affirme, une approximation exponontielle pour la distribution du temps de transfert ne devrait pas être trop mauvaise.

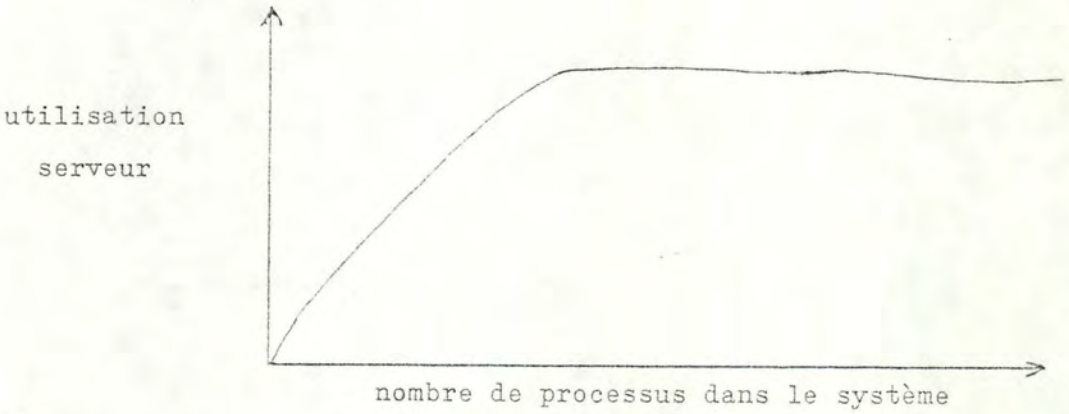
Le nombre moyen de demandes transférées de la PWQ vers la TWQ selon l'algorithme "SCAN" d'optimisation du positionnement du bras (cfr. I.5.2.) dépend fortement de la charge : plus il y a de demandes, plus on a de chances d'en avoir concernant le même cylindre. Aussi, avons nous choisi de donner pour chaque unité la fourchette obtenue par les résultats (de la même manière qu'en III.4.1. pour les nombres de processus).

PS:1	[2,00 ; 2,55]
PS:2	[1,98 ; 2,53]
PS:3	[2,04 ; 2,93]
US :	[1,97 ; 4,37]
Montable :	[2,22 ; 9,29]

Pour l'unité montable, l'écart est très grand. On pourrait peut être en conclure que : comme c'est en général une seule personne qui demande le montage d'un disque particulier sur cette unité, cette unique personne travaille sur des cylindres forts rapprochés et souvent sur le même cylindre, ce qui expliquerait aussi le temps moyen de positionnement très court (10.10 ms).

III.4.5. Pourcentage d'utilisation des serveurs

Tout comme la mesure du nombre de processus dans le "balance set" (cfr. III.4.4.), il est évident que le pourcentage d'utilisation des serveurs ("CPU", unités et canaux) dépend fortement de la charge : plus il y a de processus, plus les serveurs seront utilisés. Pourtant il doit y avoir un pourcentage maximum d'utilisation des serveurs : à partir d'un certain moment, tous les processus ne peuvent plus entrer dans le "balance set". Donc même si le nombre de processus dans le système augmente, le pourcentage d'utilisation des serveurs devrait se stabiliser. On obtiendrait une courbe du type suivant :



N.B. : on obtient le même type de courbe pour le nombre de processus dans le "balance set".

Voici les fourchettes que forment les différentes exécutions du mesureur.

- Pourcentage d'utilisation du "CPU" : 42,54 ; 52,45
- Pourcentage d'utilisation des unités.

PS:1	[21,00 ; 32,92]
PS:2	[17,11 ; 27,40]
PS:3	[19,32 ; 28,33]
US :	[1,87 ; 13,55]
montable :	[0,00 ; 0,04]

- Pourcentage d'utilisation des canaux.

CANAL 0	[14,58 ; 32,38]
CANAL 1	[12,41 ; 18,27]
CANAL 2	[12,29 ; 19,27]
CANAL 3	[4,83 ; 37,22]

III.5. PERTURBATION INTRODUITE PAR LES MESURES

La perturbation introduite par un mesureur est fonction de la durée et de la fréquence des mesures.

Ces deux données n'étant pas faciles à observer étant donné la nature de notre mesureur, nous ne nous sommes pas attardés d'avantage sur ce point, constatant de visu que le système était fiable et que la surcharge introduite n'apparaissait pas à l'utilisateur derrière son terminal.

CHAPITRE IV

RESOLUTION DU MODELE

IV.0. INTRODUCTION

Comme on le sait, un modèle de file d'attente, souvent appelé réseau, est composé de serveurs, de files d'attente pour accéder à ces serveurs et de clients qui effectuent des transitions entre ces serveurs. C'est une représentation formalisée de la compréhension que nous avons de la réalité. Pour étudier le comportement du système, nous avons choisi d'élaborer un tel modèle.

Nous avons décrit dans le chapitre II, la modélisation que nous nous proposons de faire du système. Ensuite, dans le chapitre III, nous avons expliqué la manière dont nous avons réalisé les mesures des données qui nous sont nécessaires pour résoudre le modèle et pour le vérifier. Nous allons, dans ce chapitre présenter comment nous avons résolu le modèle.
Dans un premier temps, nous présenterons les outils permettant de résoudre de tels modèles (IV.1.).
Puis, nous énoncerons une méthode, assez répandue, pour employer ces outils : la décomposition (IV.2.).
Nous décrirons les différentes solutions possibles applicables au modèle avec leurs avantages et inconvénients (IV.3.) et présenterons la résolution que nous avons adoptée (IV.4.). Enfin, nous pourrons analyser les résultats obtenus (IV.5.). Nous signalons que le programme de résolution du modèle et les résultats obtenus se trouvent en annexe.

IV.1. LES OUTILS DE RESOLUTION

La résolution d'un modèle nous fournit des indications quant au comportement de celui-ci. Elle peut nous donner par exemple la longueur moyenne des files d'attente, le temps moyen de service pour chaque serveur, le temps moyen que doit attendre un client avant d'être servi, ...

Il existe beaucoup d'outils permettant de telles analyses. Nous commencerons par décrire les différents types d'outils (IV.1.1.). Ici, à l'institut d'informatique, nous disposons de l'outil QNAP, nous le décrirons en IV.1.2. Ensuite nous donnerons un bref aperçu des autres outils existant actuellement (IV.1.3.).

IV.1.1. Les méthodes de résolution [CHSA, CORN, FICH]

Il existe différentes méthodes pour résoudre les réseaux de files d'attente. Nous les classons en deux types : les méthodes mathématiques et les méthodes recourant à la simulation. Le choix qu'un analyste peut faire parmi ces méthodes dépendra du temps d'exécution qu'elles requièrent, du degré de précision qu'elles fournissent pour ces résultats par rapport au degré de précision requis pour le problème, et enfin, et ce n'est pas le moindre critère, de la confiance, la crédibilité que l'analyste a dans l'outil.
Mais souvent ces trois critères s'opposent.

IV.1.1.1. Les méthodes mathématiques

Les méthodes de résolution mathématiques (ou analytiques) sont caractérisées par un ensemble d'équations représentant l'évolution d'un certain nombre de variables typiques du modèle. Elles permettent ainsi de prédire l'évolution du comportement du modèle. Malheureusement, ces méthodes analytiques ne permettent pas de prendre en considération tous les modèles : elles demandent un certain nombre de simplifications et d'hypothèses non réalistes.

Il existe une grande variété de ces méthodes. Certaines fournissent des résultats exacts, d'autres des résultats approximatifs. Si tous les modèles ne peuvent pas être résolus analytiquement, il y en a encore moins qui peuvent l'être par une méthode fournissant des résultats corrects. Exiger des résultats exacts coûte également très cher, demande beaucoup de temps d'exécution.

IV.1.1.2. Les simulations

Dans un simulateur, les équations sont remplacées par des algorithmes. Il ne s'agit plus de résolution arithmétique. Pendant un certain temps, on reproduit le comportement dynamique des différents composants du modèle comme dans le système réel. Il est donc possible, en simulant, de résoudre plus de modèles que mathématiquement. Cependant, un simulateur coûte cher à l'exécution.
Tout d'abord, pour obtenir des résultats corrects, il faut simuler jusqu'à ce que le modèle ait atteint l'état stationnaire. De plus, si on veut des résultats se rapprochant le plus possible de la réalité, la simulation doit s'exécuter le plus longtemps possible : plus la simulation sera longue, meilleures seront les estimations statistiques produisant les résultats.

En procédant par simulation, nous nous trouvons donc devant deux problèmes principaux : déterminer la précision des estimations obtenues par rapport aux valeurs correctes et déterminer la longueur de la simulation, afin d'obtenir des estimations assez proches des valeurs correctes.
Une approche en général adoptée pour résoudre ces problèmes est de construire un intervalle de confiance pour un résultat de la simulation du modèle. Si une simulation, pour un paramètre résultat, produit l'estimation R, on détermine l'intervalle de confiance de ce paramètre, $[R - \delta , R + \delta]$ pour un certain niveau de confiance (ex. : $\alpha = 0,1$ ou $0,05$) et on dit que l'intervalle contient dans $100 (1 - \alpha)$ % des cas la valeur vraie (r) du paramètre résultat.

C'est-à-dire

$$\Pr\left[R - \delta \leqslant r \leqslant R + \delta \right] = 1 - \alpha$$

On peut aussi interpréter cet intervalle de la manière suivante : supposons que l'on exécute la simulation d'un modèle plusieurs fois, avec des nombres différents générés aléatoirement pour chaque exécution, et que l'on calcule l'intervalle de confiance de R pour chaque exécution, on peut alors s'attendre à ce que le pourcentage des exécutions contenant la valeur exacte soit égal à $(1 - \alpha)\ 100$ (ex. : 90 ou 95 %).

Le deuxième problème peut être surmonté en utilisant des règles de "stoppage" séquentielles en combinaison avec la construction d'un intervalle de confiance. Régulièrement, après un temps déterminé d'exécution de la simulation, on calcule l'intervalle de confiance. S'il est suffisamment étroit, on arrête la simulation. Sinon on continue pendant une tranche de temps supplémentaire.

IV.1.1.3. Les autres méthodes

Il faut aussi noter qu'il existe des méthodes de résolution hybrides, c'est-à-dire des méthodes utilisant à la fois des résolutions mathématiques et des simulations. Un autre moyen de résoudre un modèle de files d'attente est d'utiliser la décomposition. Cette méthode étant assez intéressante, nous y avons consacré une partie dans ce chapitre (IV.2.).

IV.1.2. QNAP [QNAP]

QNAP (Queuing Network Analysis Package) est un de ces outils destinés à décrire, traiter et résoudre des modèles de files d'attente. Comme il était le seul outil dont nous disposions, c'est lui que nous avons utilisé. Comme nous l'avons décrit longuement dans l'annexe D, nous ne reprendrons ici que les caractéristiques principales de QNAP.

IV.1.2.1. Le langage

Le langage de QNAP nous fournit une méthode pour définir le réseau, spécifier la méthode de résolution désirée et gérer les résultats. Il est utilisable de la même manière que tout autre langage de programmation classique, c'est-à-dire qu'il ne peut être utilisé interactivement à l'aide de commandes mais est uniquement utilisable à l'intérieur d'un programme.

Le langage de description des modèles présente (en outre) les avantages suivants :

- il fournit un seul format de définition pour toutes les méthodes de résolution
- il permet de définir des réseaux ouverts, fermés ou mixtes
- il permet de définir des serveurs passifs (ou ressources) ; un client pouvant posséder plusieurs ressources simultanément
- il permet l'utilisation de "flags" et de "sémaphores" pour gérer l'accès aux serveurs
- il permet des distributions de service non-exponentielles et permet que ce service soit dépendant du nombre de clients dans la station
- il permet différentes disciplines pour les files : FIFO, LIFO, avec ou sans priorités entre les clients, avec ou sans possibilité de préemption, avec ou sans définition de "quantum".

IV.1.2.2. Les types de résolution

Grâce à QNAP, nous pouvons résoudre des réseaux de différentes manières. Nous pouvons simuler les réseaux ou utilier une des nombreuses méthodes analytiques que QNAP fournit. Dans les méthodes analytiques, il y en a qui obtiennent des résultats exacts et d'autres des résultats approximatifs. On différencie des autres méthodes analytiques, la méthode de MARKOV. Elle fournit les résultats les plus fiables (exacts) mais consomme également énormément de temps "CPU" et pour cela n'est pas très utilisée. Par simulation, nous obtiendrons toujours des résultats approximatifs. C'est une estimation des performances d'un modèle dans un régime stationnaire, mais ce régime n'est pas toujours atteint.

Il suffit à l'utilisateur de préciser s'il veut une simulation (SIMUL), une résolution par méthode de Markov (MARKOV), ou une autre résolution analytique (SOLVE). Dans ce dernier cas, c'est QNAP lui-même qui déterminera parmi les méthodes analytiques, laquelle est applicable au réseau qui a été défini. QNAP permet également à l'utilisateur de sélectionner un sous-modèle du modèle global pour le résoudre.
Le "solver" choisi effectue en premier lieu l'évaluation des paramètres de chaque station du réseau (probabilité de transition, nombre initial de clients, nombre de serveurs, niveau de priorité...). Ensuite, si c'est "SOLVE" qui a été demandé, il évalue le service. Tandis que si c'est "MARKOV" ou "SIMUL" qui ont été sollicités, le service est exécuté dynamiquement pendant la résolution.

Examinons maintenant les principales restrictions et les principaux avantages de chacune de ces trois méthodes :

1) résolutions analytiques : SOLVE

Ces résolutions permettent des probabilités de transitions différentes par classe de clients, différentes distributions pour les services ; mais n'autorisent pas les serveurs passifs (ou ressources), les "sémaphores" et les "flags".

Le service d'une station ne peut être constitué que d'une simple demande de travail, et ne peut pas dépendre de l'état du modèle. De plus, elles imposent certaines restrictions quant au type de réseaux (ayant trait aux réseaux ouverts, sous-chaines ouvertes, réseaux mixtes, stations sources) (cfr. annexe D). En plus de ces restrictions globales, chaque méthode de résolution particulière possède les siennes propres.

2) la résolution markovienne (MARKOV)

Cette résolution fournit des résultats exacts mais avec un temps d'exécution très long et pour un nombre de réseaux très limité. Cette résolution est applicable uniquement aux modèles ayant un nombre fini d'états. Elle s'applique uniquement aux réseaux fermés contenant une ou plusieurs classes de clients. Elle n'autorise pas les stations "ressource" et "sémaphore". Elle s'applique en particulier aux modèles "markoviens" c'est-à-dire aux modèles dont toutes les distributions de service sont exponentielles et dont les disciplines sont "work conserving".

La résolution s'effectue selon les étapes suivantes :

- vérifie que le modèle peut être résolu par MARKOV.
- détermine tous les états possibles du modèle et leurs probabilités de transitions
- détermine les probabilités en régime stationnaire de chacun d'entre eux
- calcule les performances de chaque composant du modèle.

3) la simulation (SIMUL)

La simulation s'applique à pratiquement tous les réseaux. QNAP nous fournit certaines facilités pour déterminer la longueur de simulation et la précision des résultats.

- on détermine soi-même la longueur maximum de simulation
- on peut exécuter périodiquement pendant la simulation une séquence de test
- on peut stopper soi-même la simulation à n'importe quel moment (même si la longueur maximum n'a pas été atteinte).
- on peut obtenir dans le rapport, pour certaines stations, l'intervalle de confiance des résultats.
- on peut éclater la simulation en plusieurs blocs successifs et indépendants
- on peut tester l'hypothèse d'indépendance entre ces blocs par la fonction d'"autocorrélation".

IV.1.2.3. Les résultats

A la fin de la résolution, les résultats sont imprimés dans un rapport standard. Les différents résultats imprimés dans ce rapport sont pour chaque station définie :

- le facteur d'utilisation
  c'est-à-dire le pourcentage de temps pendant lequel le serveur (ou la ressource) est occupé
- le temps moyen de service
  c'est-à-dire le temps moyen de service tel que le voit le client
- le temps moyen d'attente
  c'est-à-dire la somme du temps d'attente et du temps de service
- le nombre moyen de clients dans la station
- le "throughput" moyen
  c'est-à-dire le nombre moyen de clients servis par unité de temps.

L'utilisateur de QNAP peut demander l'impression complète de tout ce rapport, ou il peut demander à disposer de certains résultats du rapport.

N.B. : pour la simulation, ce n'est pas le "throughput" qui est produit dans le rapport mais bien le nombre de clients servis par la station.

IV.1.3. Les autres outils [AGRA, CHSA]

Sur le marché, il existe beaucoup de systèmes permettant de résoudre les modèles. Parmi les plus connus, citons RESQ, CADS, BEST/1, THEsolver et SuperNet.

CADS, BEST/1, THE solver et SuperNet permettent des résolutions analytiques exactes et approximatives. Tandis que RESQ permet.des résolutions analytiques, par simulation, hybrides (mixage des deux autres) et par décomposition (décrite en IV.2.).

De plus, il fournit des méthodes calculant l'intervalle de confiance et implémentant des règles de "stoppage" séquentielles.

Nous allons maintenant donner une idée des possibilités d'un outil très performant : STEP-1.

STEP-1 est un outil de résolution de réseau très élaboré, qui a été implémenté aux Etats-Unis à l'université de Maryland. Par rapport à d'autres outils, il présente de nombreux avantages.

1) Les méthodes de résolution

STEP-1 propose à l'utilisateur plusieurs techniques de résolution des modèles :

- analytique exacte
- analytique approximative
- simulation
- hybrides (mixage de solution analytique et de simulation)
- par décomposition (méthode expliquée en IV.2.).

Cependant, malgré cette diversité apportée, ces méthodes deviendront un jour dépassées. Aussi, STEP-1 permet-il à l'utilisateur d'ajouter de nouvelles techniques de résolution et pour cela lui fournit des facilités.

2) L'interface avec l'utilisateur

STEP-1 est "user-friendly", il permet à l'utilisateur de résoudre les réseaux de manière interactive. Le but de l'interface avec l'utilisateur est de fournir un moyen d'interaction qui accomode à la fois les novices (qui ne sont pas familiés avec l'organisation et les fonctions de l'outil) et les utilisateurs expérimentés (qui souhaitent manipuler un minimum de commandes).
Pour cela, il permet à l'utilisateur de choisir entre plusieurs modes d'interaction : le mode menu, le mode commande ou le mode d'assistance ("Tutored mode"). L'utilisateur entre d'abord dans le mode menu ; il peut y rester ou aller vers le mode commande. Le mode d'assistance est accessible à partir des autres modes, il permet de décrire les différents paramètres d'une commande et leurs valeurs par défaut. L'utilisateur peut rester dans ce mode pour donner une valeur à ces paramètres, il n'est pas obligé de retourner dans un des deux autres modes. Le mode assistance est donc plus qu'une commande traditionnelle "help".

IV.2. METHODE DE DECOMPOSITION [BLUM, CHSA, NOIG, NOIT]

La méthode considérée comme la plus importante pour résoudre des modèles de manière approximative est la décomposition. On résoud, de manière isolée, certaines portions du modèle et on rassemble les résultats pour produire la solution du modèle entier. En d'autres termes, à partir du modèle à résoudre, on détermine un sous-modèle (parfois plusieurs). On le résoud séparément. Enfin, on résoud le modèle agrégé, c'est-à-dire le modèle de départ dans lequel le sous-modèle est remplacé par un seul serveur avec une seule file. Le service de ce serveur est déterminé par la résolution du sous-modèle et dépend en général de la charge du sous-modèle. De cette manière, on remplace donc le sous-modèle par un autre plus simple. Cette approche est aussi souvent appelée agrégation.

IV.2.1. Exemple

Pour mieux comprendre ce qu'est la décomposition, prenons un exemple. La figure IV.1. représente un modèle composé d'un processeur central et d'une file pour accéder aux processeurs périphériques (PP).
Pour effectuer une Entrée/Sortie, le processus a besoin à la fois d'un processeur périphérique et d'un disque. Or il y a 4 processeurs périphériques.

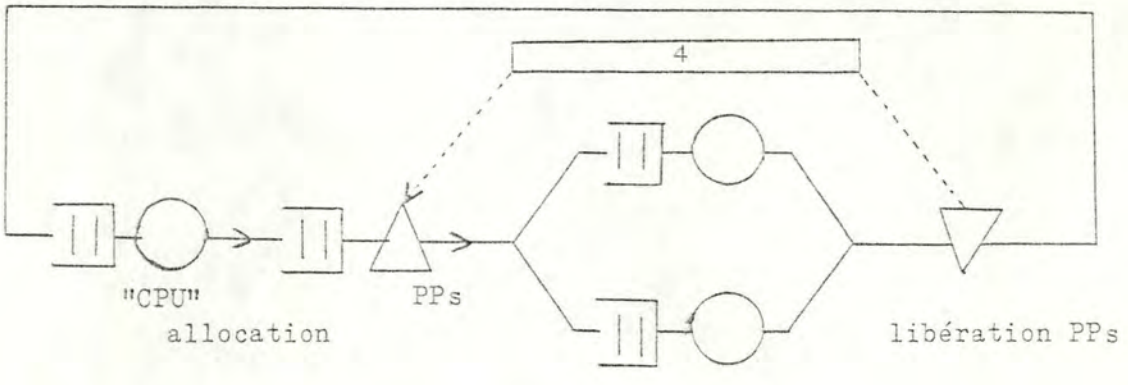


Figure IV.1.

Quand un processus termine son service dans le "CPU", il joint la file des PPs. C'est seulement quand il peut obtenir un PP qu'il effectue son E/S. Après cela, il libère le PP et entre dans la file pour être servi par le "CPU".

Nous nous proposons de construire un sous-système avec la file PP et les disques. Les figures IV.2. et IV.3. représentent le sous-modèle et le modèle agrégé. Pour résoudre le sous-modèle, nous connectons sa sortie à son entrée, créant ainsi une boucle de réalimentation.

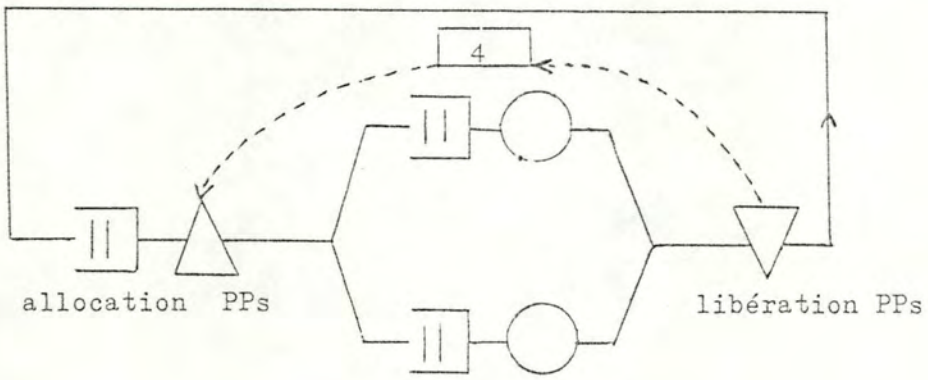


Figure IV.2.

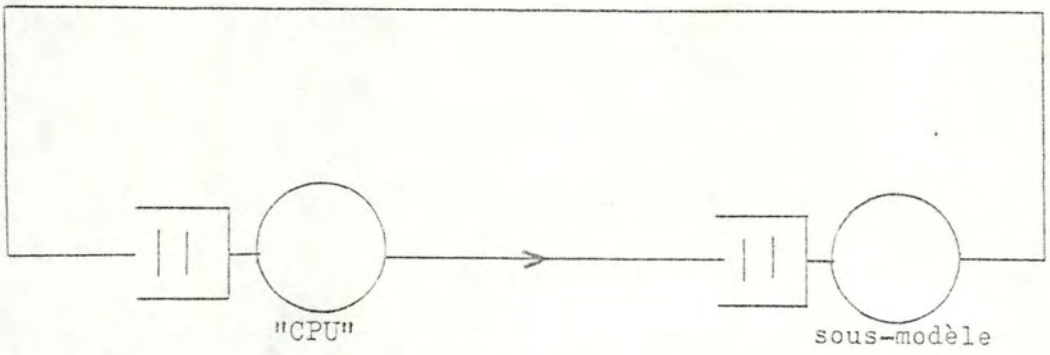


Figure IV.3.

Nous allons déterminer le "throughput", c'est-à-dire le nombre de processus passant par la boucle de réalimentation par unité de temps : X(n), et cela pour chaque grandeur de population possible (n) dans le sous-modèle.

Le temps de service moyen du serveur remplaçant le sous-modèle dans le modèle agrégé : S(n) est alors déterminé pour chaque valeur de n par 1/X (n). Dans le cas présent, on calcule X(n) pour n = 1, 2, 3, 4. Pour n supérieur à 4, X(n) sera égal à X(4) vu qu'il ne peut pas y avoir plus de 4 processus dans le sous-modèle.

De cette manière, on a remplacé le sous-modèle par un serveur dont le service dépend de la charge du sous-modèle.

IV.2.2. Avantages et désavantages d'une telle méthode

L'avantage que présente cette méthode, est de permettre une économie de coût d'exécution de la résolution. Dans le cas où le modèle à résoudre n'est pas conforme à la résolution analytique, la décomposition peut amener à un sous-modèle et/ou un modèle agrégé conformes. Or, on le sait, une simulation est très coûteuse par rapport à une résolution analytique. Les expériences prouvent que dans les cas où, soit le sous-modèle soit le modèle agrégé (ou même les deux) doivent être simulés, la décomposition reste une approche attrayante. En effet, on remplace par un seul événement (sous-modèle) ce qui par la simulation directe aurait nécessité plusieurs événements.

Mais cette méthode comporte également des désavantages. Premièrement elle n'implémente pas toutes les interactions qui pourraient exister entre les sous-modèles et le modèle agrégé. La décomposition est une méthode approximative et ne fournit donc que des résultats approximatifs pour le modèle global. Deuxièmement, la décomposition n'estime que des moyennes et pas des distributions.

Troisièmement, des résultats ne sont générés que pour le modèle agrégé et pas pour le sous-modèle. Si nous voulons des résultats pour une certaine file, celle-ci doit apparaître explicitement dans le modèle agrégé, ou alors il faut les calculer soi-même.

IV.2.3. Enseignements tirés des expériences réalisées à ce sujet

Le centre de recherche "IBM, Thomas J. Watson" a mené diverses expériences pour déterminer l'efficacité d'une telle méthode. Ces expériences portent sur les trois types de décomposition suivants :

- modèle agrégé résolu par simulation
  sous-modèle résolu analytiquement
- modèle agrégé résolu analytiquement
  sous-modèle résolu par simulation
- modèle agrégé résolu par simulation
  sous-modèle résolu par simulation.

  et pour ce cas, ils allouent au sous-modèle différents poucentages du temps total de simulation (5, 25, 50, 75 et 95 %).

Ils exécutent chaque expérience 20 fois et construisent pour chaque résultat intéressant un intervalle de confiance de 90 pourcent. Enfin, ils comparent les intervalles obtenus par décomposition avec ceux obtenus par simulation directe.

En général, on peut affirmer que la décomposition où l'un des deux modèles (sous-modèle ou modèle agrégé) est résolu analytiquement est favorable : l'intervalle de confiance obtenu est beaucoup moins large que celui obtenu par simulation. Pour produire la même précision, une simulation directe devrait s'exécuter au moins 100 fois plus longtemps. Cette différence de précision augmente encore si le nombre d'événements à l'intérieur du sous-modèle augmente par rapport au nombre d'événements à l'intérieur du modèle agrégé.

Dans le cas où la décomposition amène à résoudre les deux modèles. par simulation, les conclusions sont moins claires. Toutefois on peut dire que la meilleure répartition de temps de résolution entre les deux modèles est de 50 % - 50 %. On remarque aussi que si le nombre d'événements à l'intérieur du sous-modèle est élevé, on a de meilleurs résultats en décomposant (sim - sim) que par simulation directe. Dans le cas contraire, c'est la simulation directe qui est favorisée.

A partir de là, le groupe a tiré des règles spécifiant les cas où la décomposition est applicable et d'autres où elle est inappropriée.

IV.2.4. Règles d'"applicabilité"

Il y a de nombreux cas où la décomposition est bénéfique.

1) quand la décomposition amène à un sous-modèle, ou un modèle agrégé qui peut être résolu de manière analytique.
2) Quand l'étude d'un modèle est telle que les paramètres à fournir au sous-modèle ne changent jamais. Il suffit dès lors, de résoudre une seule fois le sous-modèle et de fournir les résultats au modèle agrégé.
3) Quand les échelles de temps d'occurence des événements dans les deux modèles sont très différentes.
4) Quand un modèle contient plusieurs sous-modèles identiques.
5) Quand on a à faire à une grande simulation qui requiert beaucoup de ressources, la décomposition réduit la quantité de mémoire nécessaire pour résoudre le modèle.

IV.2.5. Règles d'"inapplicabilité"

Dans certaines circonstances, la décomposition est inappropriée. Pourtant, une reconception astucieuse du modèle peut souvent surmonter ces problèmes.

1) Quand un sous-modèle est composé de plusieurs points d'entrée et de sortie.
2) Quand les états d'un des deux modèles dépendent des états de l'autre modèle.
   Par exemple, quand les événements du sous-modèle doivent être synchronisés avec ceux du modèle agrégé.
3) Quand le nombre d'occurences d'événements dans le sous-modèle n'est pas largement supérieur au nombre d'occurences d'événements dans le modèle agrégé.
4) Quand le sous-modèle contient plusieurs classes fermées.

IV.2.6. Décomposabilité presque complète

Toutes les règles d'"applicabilité et d'"inapplicabilité" que nous venons d'énoncer sont tirées d'expériences. Quand on applique ces règles pour décomposer un modèle, on obtient des résultats approximatifs : l'expérience prouve que les résultats obtenus se rapprochent des résultats exacts.

Cependant, il existe des théorèmes sur la décomposabilité des réseaux. Essayons de les synthétiser.

* Décomposabilité presque complète.
  Si des sous-modèles d'un même modèle interagissent peu entre eux (en comparaison avec les interactions qui existent à l'intérieur de ces sous-modèles), on peut d'une part étudier les sous-modèles sans s'occuper des interactions entre eux et d'autre part étudier les interactions entre les sous-modèles sans s'occuper des interactions à l'intérieur des sous-modèles. De tels systèmes sont appelés presque complètement décomposables.

* Le serveur de Jackson

  Un serveur de Jackson est un serveur exponentiel dont le taux est fonction du nombre de clients dans la file.

  Un sous-modèle fermé peut être assimilé à un tel serveur et le taux (dépendant du nombre de clients dans la file) peut être calculé.

Ces théorèmes ont été démontrés dans la littérature, mais ce n'est pas notre but ici d'expliquer ces démonstrations.

Nous en déduisons donc que si un système est presque complètement décomposable, on peut le résoudre en deux temps.

1) résoudre les sous-modèles
2) résoudre le modèle où chaque sous-modèles est remplacé par un serveur unique dont le service dépend du nombre de clients dans ce sous-modèle.

C'est-à-dire que si on applique le principe de décomposition à un modèle dont les sous-modèles sont très peu dépendants, on obtient des résultats exacts.

IV.3. DIFFERENTES RESOLUTIONS POSSIBLES

Nous allons exposer ici différentes possibilités de résolution du modèle, employant la décomposition ou pas. Pour chaque solution, nous montrerons les contraintes qui y sont liées et en dégagerons les avantages et inconvénients.

IV.3.1. Résolution globale

La première idée est bien sûr de résoudre le modèle globalement, sans appliquer la décomposition. La figure IV.4. représente le modèle que nous voulons résoudre. Il s'agit du modèle donné en II.2.1. mais compte tenu des changements apportés des mesures (III.2.3.).

Dans ce modèle, nous devons absolument tenir compte de 2 faits assez importants dans le système :

- d'une part, nous devons tenir compte de la gestion du "balance set" (pour des détails, cfr. I.3.2.), celle-ci se manifeste dans notre modèle par la présence d'une file d'attente d'allocation de mémoire centrale ainsi que des triangles droits et renversés ,
- d'autre part, nous voulons aussi que notre résolution prenne en compte le fait que, lorsque une unité s'est positionnée, elle reste bloquée durant le transfert (autrement dit, un serveur disque reste bloqué durant le positionnement et le transfert par le serveur canal qui lui est associé).

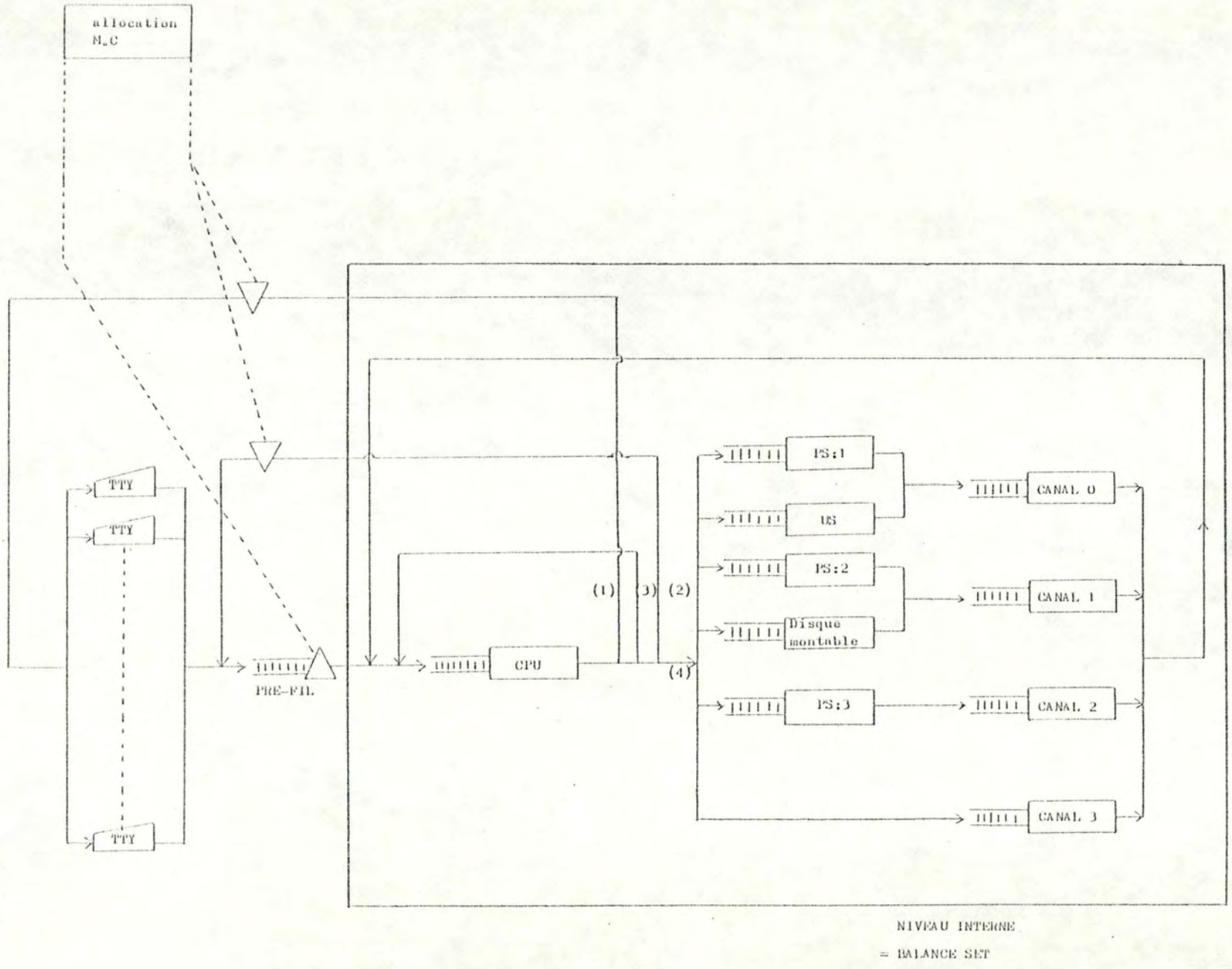

allocation
M.C
TTY
TTY
TTY
PRE-FIL
CPU
(1)
(3)
(2)
(4)
PS:1
US
PS:2
Disque
montable
PS:3
CANAL 0
CANAL 1
CANAL 2
CANAL 3
NIVEAU INTERNE
= BALANCE SET

Figure IV.4.

Or, QNAP nous fournit 2 facilités grâce auxquelles nous pouvons aisément prendre en considération ces 2 problèmes dans notre résolution : il s'agit des "flags" et des "sémaphores" (Voir IV.1.2.1.) et (annexe D). Malheureusement, aucune des méthodes analytiques (que ce soit MARKOV ou un autre SOLVER) n'accepte l'utilisation des "flags" ou des "sémaphores". Les différentes résolutions analytiques se montraient donc trop restrictives pour notre modèle. Nous devons donc nous tourner vers une simulation du modèle.

IV.3.2. Première décomposition

Nous pourrions aussi appliquer la méthode de résolution utilisant la décomposition (cfr. IV.2.) et tenter de résoudre de manière analytique le sous-modèle choisi ou le modèle agrégé obtenu.

La première manière serait de décomposer le modèle selon les 2 niveaux interne et externe (cfr. II.2.2.1.).
Le niveau interne constituerait ainsi le sous-modèle (fig. IV.5.a.) comprenant donc le "CPU", les unités et les canaux, chacun précédé de sa propre file d'attente. Mais à cela, il faut ajouter une boucle (*) reliant la sortie du sous-modèle à son entrée (cfr. exemple IV.2.1.). On obtient ainsi un sous-modèle fermé.

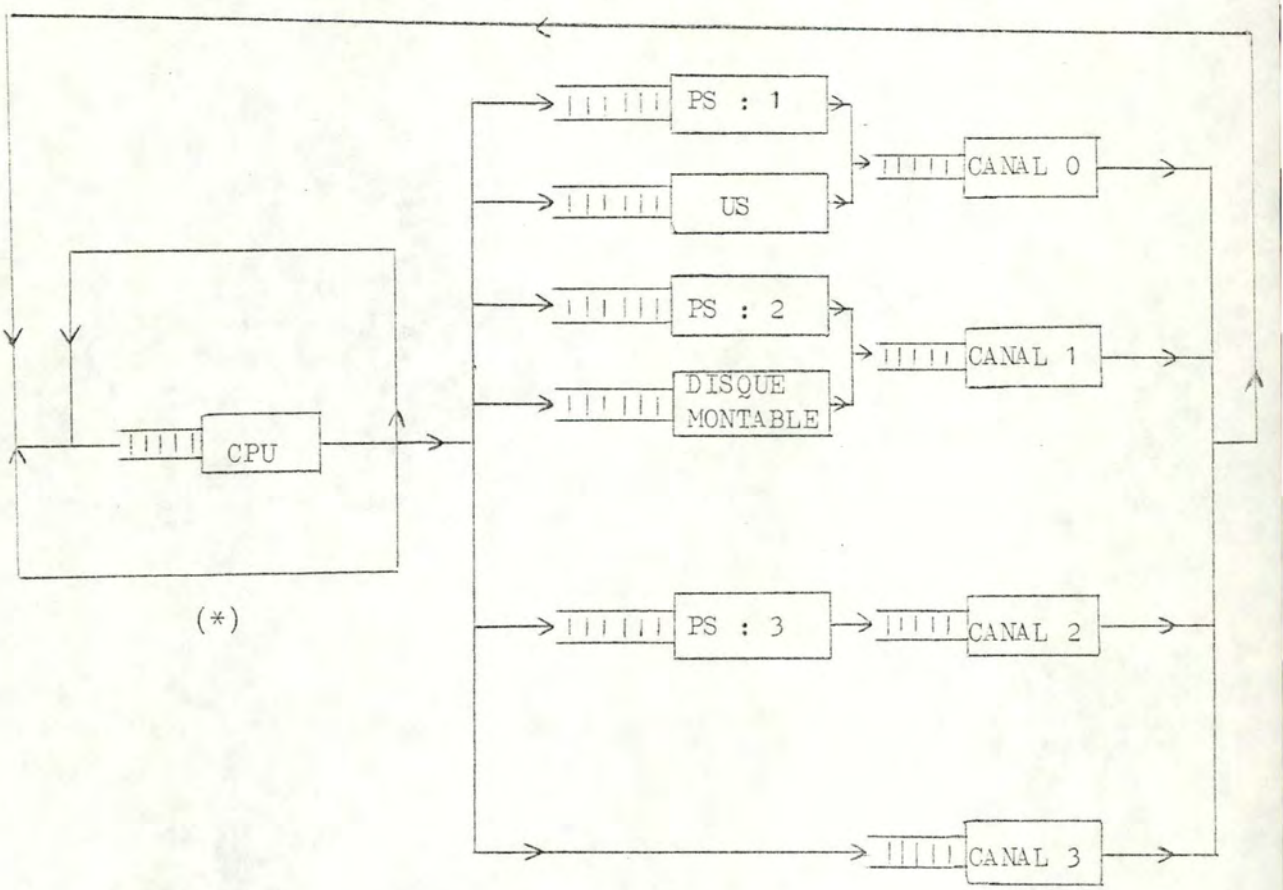


fig. IV.5.a. : SOUS-MODELE

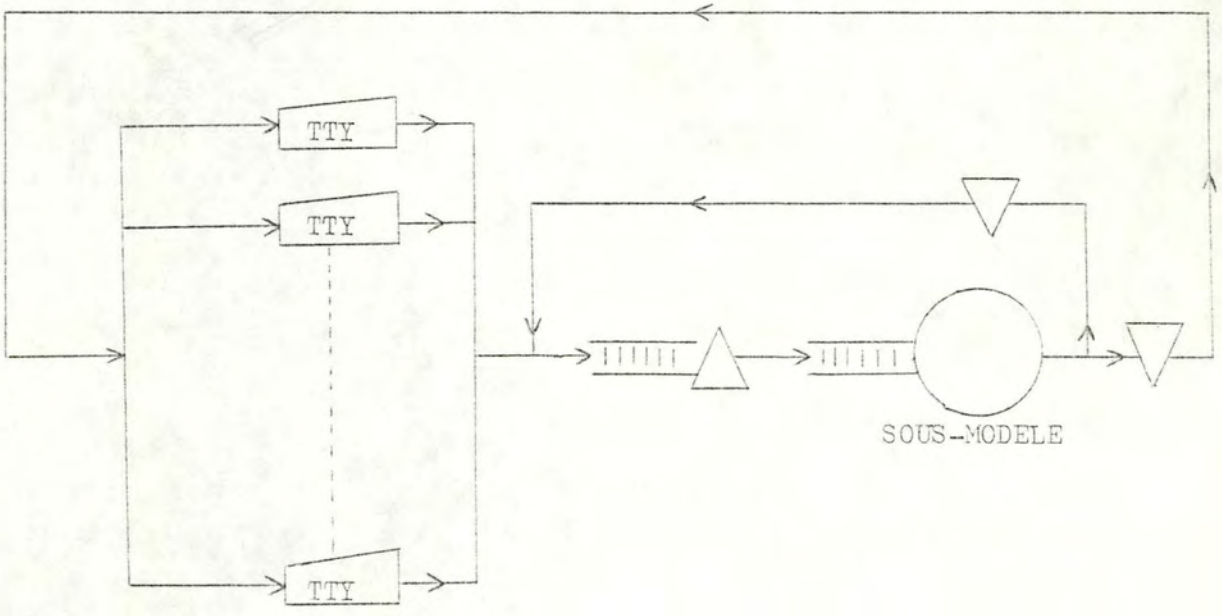


fig. IV.5.b. : MODELE AGREGE

Le modèle agrégé (fig. IV.5.b.) comprendrait le niveau externe, à savoir les différents serveurs terminaux et la file d'attente d'allocation de mémoire (PRE-FIL), ainsi qu'un simple serveur représentant le sous-modèle.

Cette décomposition applique parfaitement la théorie de la décomposabilité presque complète (cfr. IV.2.6.) : les interactions à l'intérieur du sous-modèle sont très élevées par rapport aux interactions entre le sous-modèle et le modèle agrégé. En effet, le temps de service du "CPU", des unités et des canaux est beaucoup plus petit que le temps de service des terminaux et de plus à la sortie du "CPU", un processus se dirige le plus souvent vers les unités (donc reste dans le sous-modèle). De cette manière, nous aurions dû obtenir de bons résultats. Malheureusement, QNAP ne nous permet pas de résoudre un des deux modèles de manière analytique : la PRE-FIL et la synchronisation entre disques et canaux doivent être résolus par simulation. Or, une des règles "d'applicabilité" (cfr. IV.2.4.) tirée des expériences est qu'un sous-modèle ou le modèle agrégé soit résolu analytiquement. Si les deux doivent être simulés, il n'y a aucun intérêt à une décomposition. Nous n'envisagerons donc pas cette solution.

IV.3.3. Deuxième décomposition

Une autre manière de procéder serait de prendre comme sous-modèle uniquement les différents serveurs que sont les unités et les canaux avec leurs files d'attentes respectives, auxquels on a ajouté la boucle (*) pour obtenir un sous-modèle fermé. Le sous-modèle représenterait donc les Entrées/Sorties courtes sur disques et bandes (fig. IV.6.a.). Le modèle agrégé (fig. IV.6.b) serait constitué des autres serveurs, à savoir les terminaux et le "CPU" (avec leur file d'attente), ainsi que de la file d'attente d'allocation de mémoire centrale et du serveur d'E/S courtes qu'est le sous-modèle (avec sa file d'attente).

Il est à noter que dans cette solution, le "CPU" se trouve dans le modèle agrégé, et que toute la gestion du "balance set" (allocation et désallocation de la mémoire centrale aux processus) se fait également dans le modèle agrégé (ceci est représenté par les triangles dans le schéma).

Cette décomposition n'applique pas la théorie de la décomposabilité presque complète (cfr. V.2.6.) : le sous-modèle est très dépendant du modèle agrégé, les interactions sont nombreuses. De plus, ni le modèle agrégé, ni le sous-modèle ne peuvent être résolus analytiquement par QNAP (l'un comprend la gestion du "balance set" et l'autre la synchronisation des disques et bandes). Cette décomposition n'est donc pas très intéressante à prendre en considération.

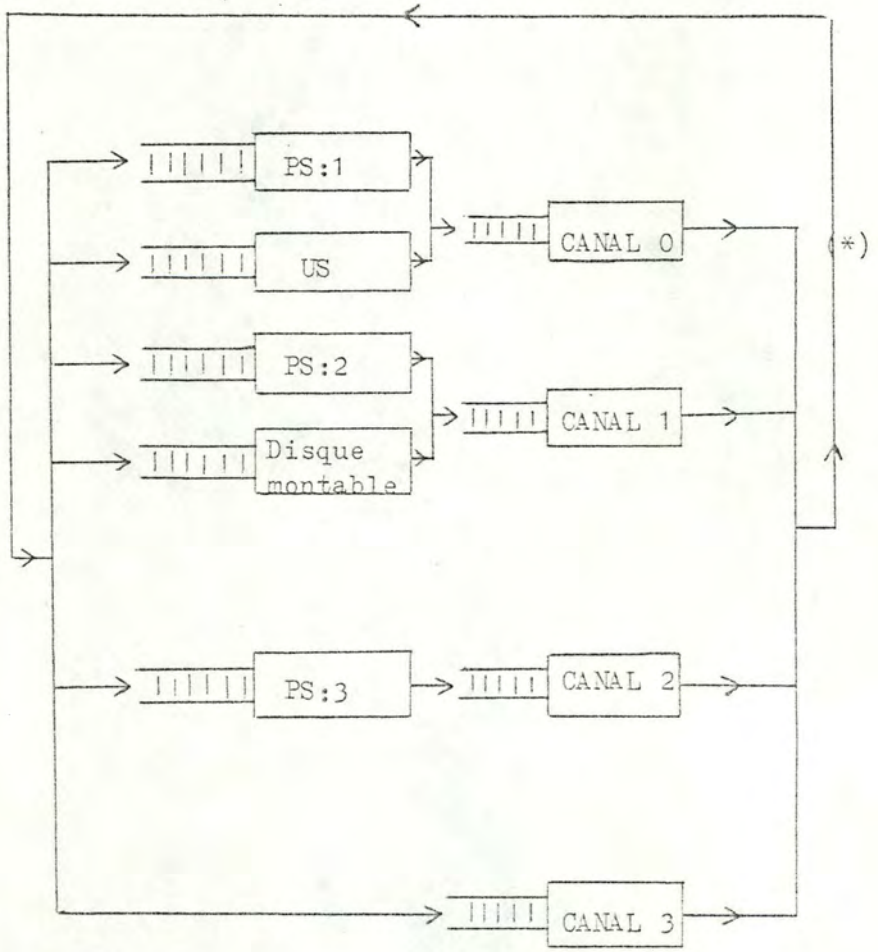


fig. IV.6.a. : SOUS-MODELE

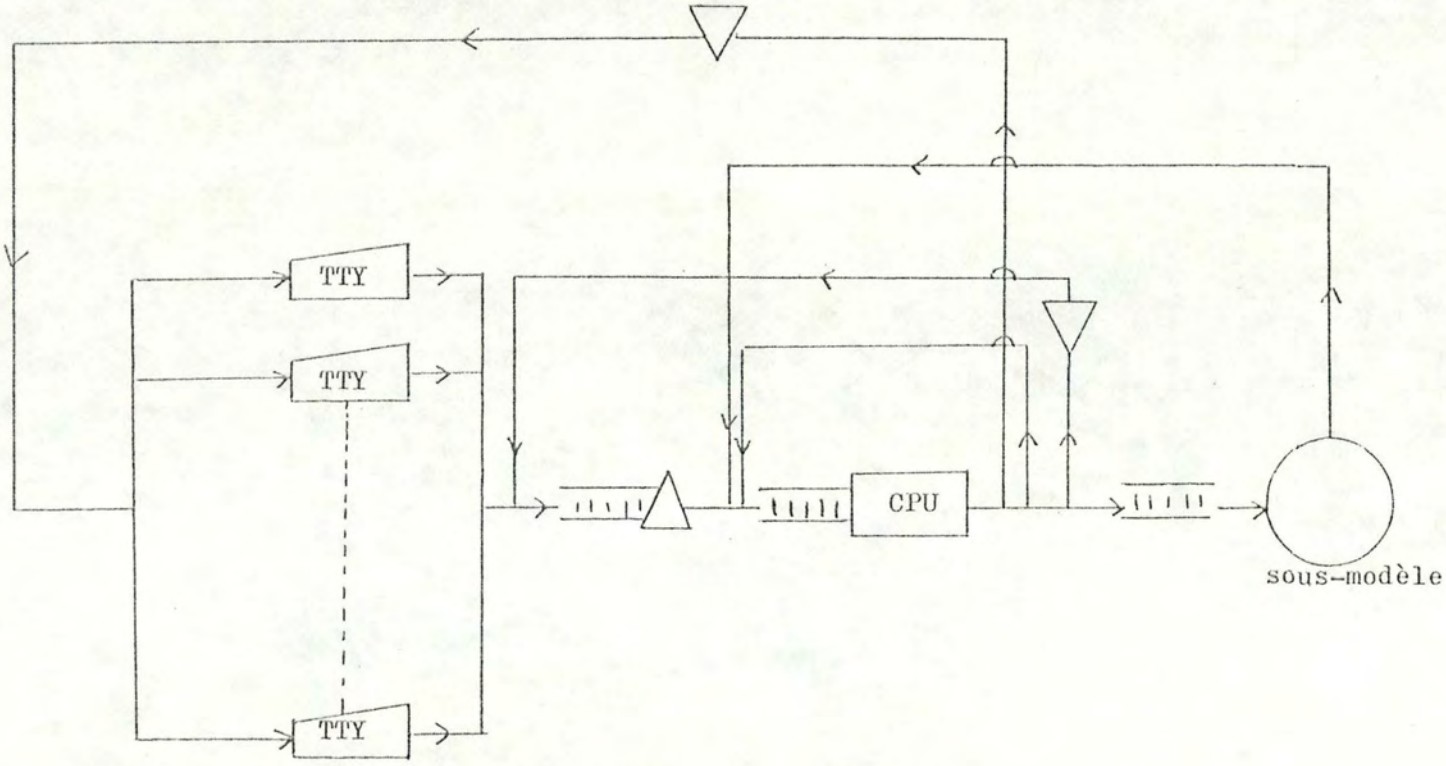


fig. IV.6.b. : MODELE AGREGE

IV.3.4. Troisième décomposition

Une troisième solution utilisant la décomposition consisterait à former 2 sous-modèles identiques composés respectivement des unités PS : 1, US et du canal 0 (fig. IV.7.a.) et des unités PS : 2, disque montable et du canal 1 (fig. IV.7.b.). A ces deux sous-modèles identiques s'en ajouterait un autre : celui réunissant l'unité PS : 3 et le canal 2 (fig. IV.7.c.). Ces trois sous-modèles, grâce aux boucles (*) sont des modèles fermés. Le modèle agrégé (fig. IV.7.d.) serait donc le modèle global de la figure IV.4. où les sous-modèles sont remplacés par des serveurs simples.

Cette décomposition n'est qu'une variante de la précédente. Cependant, si les deux premiers sous-modèles étaient identiques, on n'en simulerait qu'un seul et on pourrait en obtenir des résultats pour deux serveurs simples (cfr. IV.2.4., 4). L'avantage de cette décomposition par rapport à l'autre est de permettre la modification facile de la configuration des E/S d'un canal.

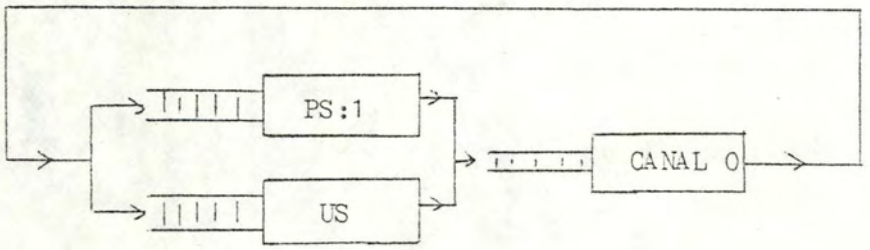


fig. IV.7.a. 1er SOUS-MODELE

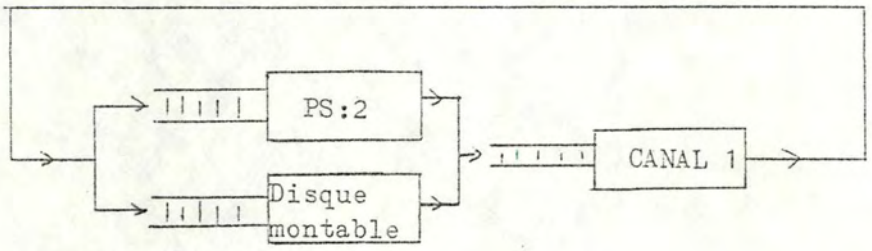


fig. IV.7.b. 2ème SOUS-MODELE

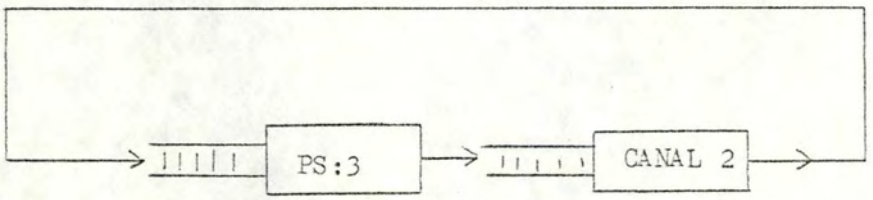


fig. IV.7.c. 3ème SOUS-MODELE

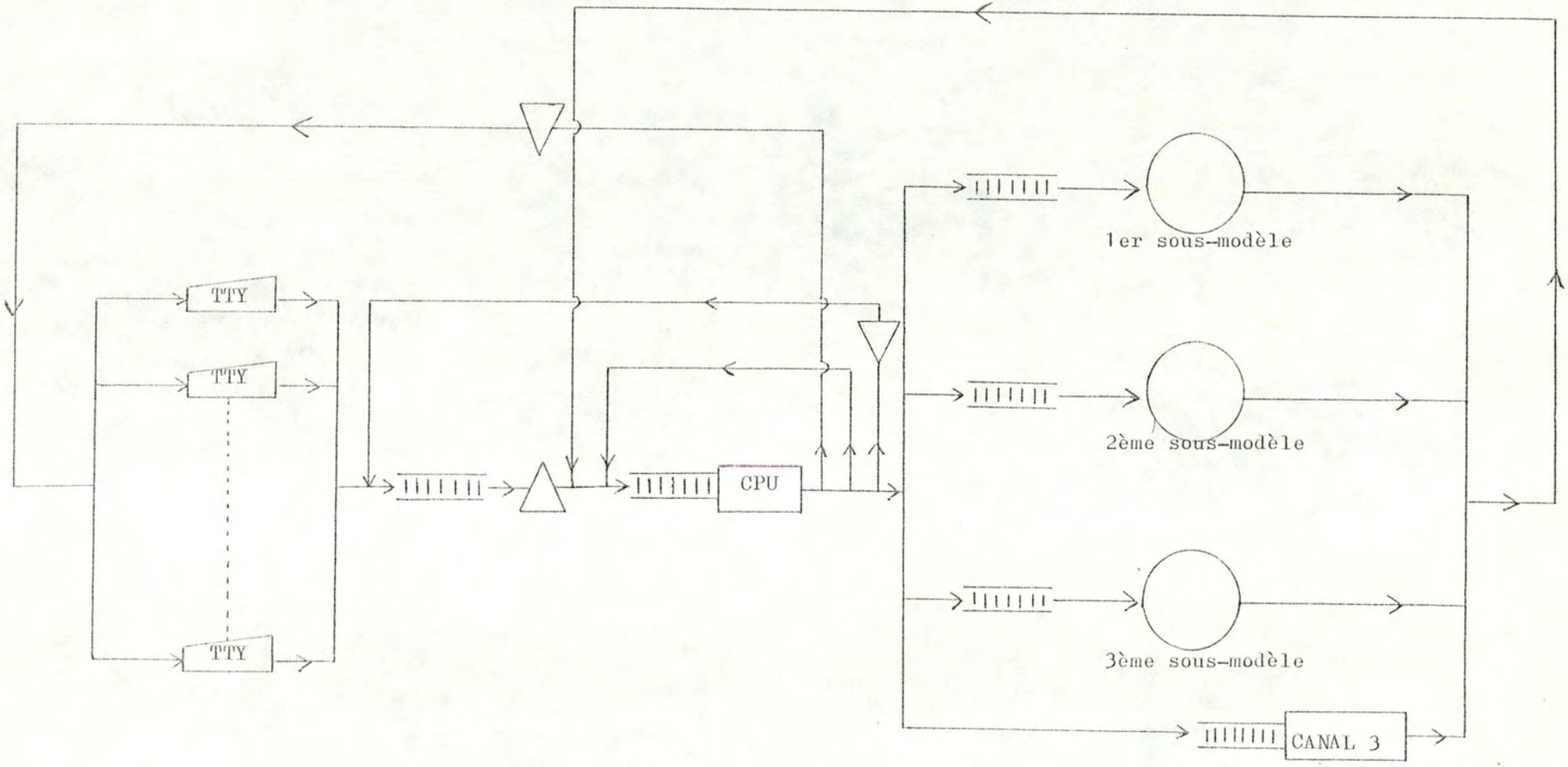


fig. IV.7.d. : MODELE AGREGE

IV.3.5. Quatrième décomposition

Nous pouvons encore envisager une quatrième décomposition. Elle est un "mixage" de la première et de la troisième et possède 3 niveaux de décomposition.

Au premier niveau, nous retrouvons les trois sous-modèles reprenant les Entrées/Sorties sur les canaux 0, 1 et 2 (fig. IV.8.a., b., c.).

Au deuxième niveau, ces trois sous-modèles sont agrégés dans un autre sous-modèle (fig. IV.8.d.). Ce sous-modèle agrégé est donc composé des serveurs simples remplaçant les trois sous-modèles précédents, du canal 3, et du "CPU". A cela s'ajoute une boucle (*) joignant la sortie du sous-modèle à son entrée.

Au troisième niveau, nous retrouvons le modèle agrégé comprenant le serveur simple remplaçant le sous-modèle, la gestion du "balance set" ($\triangle$ $\triangledown$) et les terminaux (fig. IV.8.e.).

Comme cela a déjà été remarqué pour les décompositions précédentes, le premier niveau respecte la théorie de la décomposabilité presque complète, par contre, le troisième ne la respecte pas. Les premiers sous-modèles doivent être résolus par simulation (à cause de la synchronisation entre les unités et les canaux), tandis que le sous-modèle agrégé peut être résolu analytiquement. Par contre, le modèle agrégé doit de nouveau être simulé (à cause de $\triangle$ $\triangledown$).

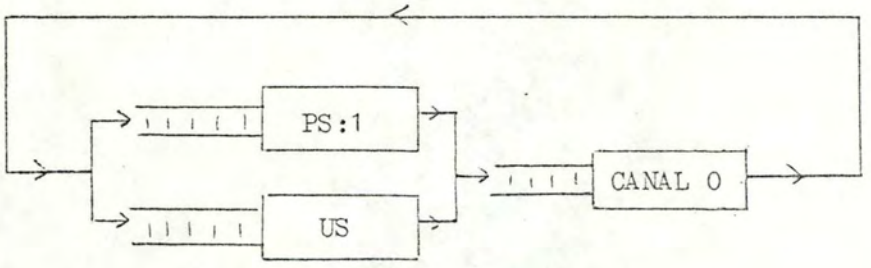


fig. IV.8.a. : 1er SOUS-MODELE

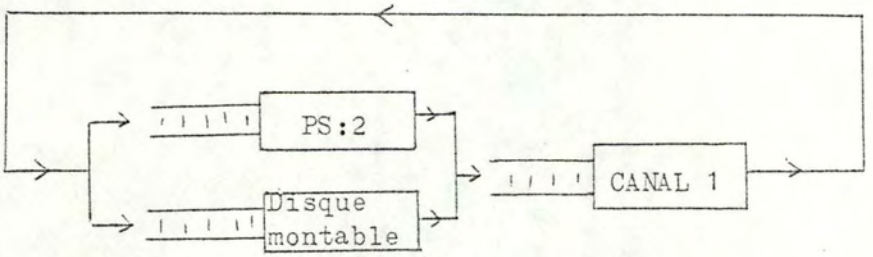


fig. IV.8.b. : 2ème SOUS-MODELE

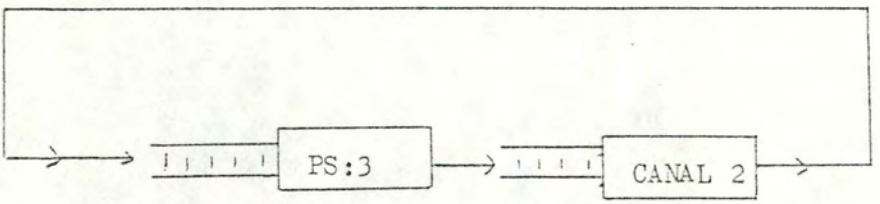


fig. IV.8.c. : 3ème SOUS-MODELE

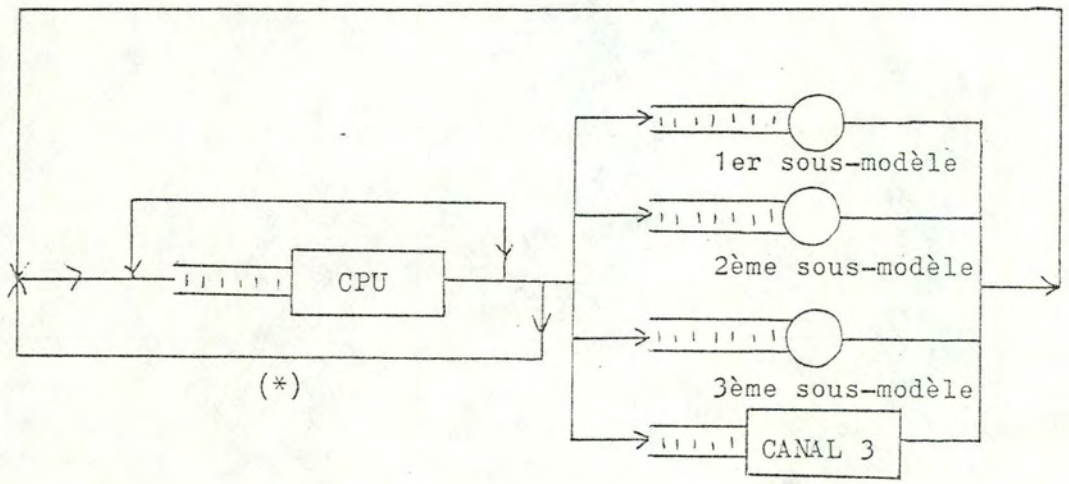


fig. IV.8.d. : SOUS-MODELE AGREGE

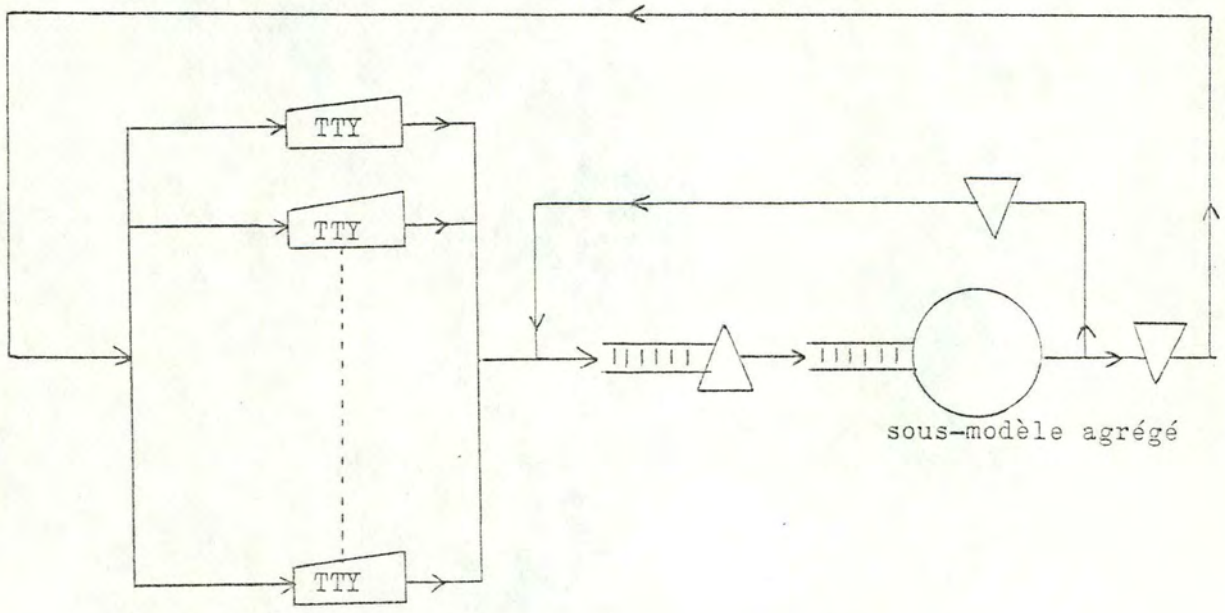


fig. IV.8.e. : MODELE AGREGE

Cette décomposition étant la seule permettant une résolution analytique, elle aurait pu être intéressante à envisager (tout en sachant que si les sous-modèles d'E/S ne respectent pas la décomposabilité presque complète, nous n'obtiendrons pas de résultats très précis). Malheureusement, QNAP ne permet pas une résolution analytique où le service d'un serveur dépend du nombre de clients dans sa file (ce qui est nécessaire dans une décomposition). La seule décomposition permettant des résolutions analytiques permises par QNAP serait celle où le sous-modèle peut être résolu analytiquement et le modèle agrégé par simulation. Or cette décomposition ne se présente pas pour notre modèle.

IV.4. IMPLEMENTATION DU MODELE

Après avoir examiné les différents outils et méthodes de résolution, et après avoir passé en revue les différentes possibilités de résolution appliquant la décomposition ou pas, nous pouvons maintenant déterminer la manière dont nous allons résoudre et implémenter le modèle avec le langage QNAP. Comme nous l'avons vu dans le point précédent, QNAP est très exigent en ce qui concerne les résolutions analytiques. Nous ne pouvons résoudre analytiquement

- ni l'allocation de la mémoire centrale exigeant l'emploi de "sémaphores" ou de "flags"
- ni la synchronisation entre canaux et unités exigeant l'emploi de "flags"
- ni l'aggrégation d'un sous-modèle dans un modèle analytique, exigeant l'emploi de distributions de service dépendant du nombre de processus dans la file du serveur (représentant le sous-modèle).

A cause de toutes ces limitations, il nous reste un seul moyen pour résoudre notre modèle de manière intéressante : la simulation de tout le modèle pris globalement (cfr. IV.3.1. et figure IV.4.).

IV.4.1. La contrainte de la charge

Comme nous l'avons dit précédemment (cfr. III.4.), nous aurions voulu étudier les mesures produites par notre mesureur par rapport à la charge du système.

Malheureusement, la charge très faible à l'époque des vacances nous a contraints à laisser tomber cette possibilité. Si nous avions pu le faire, nous aurions déterminé pour chaque mesure, une fonction analytique dépendant de cette charge. De cette manière, le seul paramètre fourni au modèle aurait été le nombre de processus présents dans le réseau.

A ce paramètre, obligatoire, auraient du s'en ajouter d'autres, facultatifs. Ces derniers auraient permis à l'utilisateur du modèle d'examiner les performances d'un système modifié. Citons quelques modifications possibles :

- Augmentation de la taille mémoire disponible pour les utilisateurs du système. Le paramètre du modèle à modifier serait ici le nombre moyen de processus dans le "balance set".
- Changement de disques ou de canaux pour des appareils plus performants. On devrait ici changer le temps moyen de positionnement des disques et le temps moyen de transfert des canaux.
- Augmentation du nombre de terminaux. Il faudrait alors augmenter le nombre de processus dans le système.
- Si l'on veut modifier le type des utilisateurs cela est très délicat. En effet il faudrait alors revoir tous les paramètres, car tout en dépend : répartitions entre les différentes classes, temps passé au terminal, probabilités de transition, temps de positionnement des disques, temps de transfert des canaux.
- Si l'on veut modifier la structure des disques, des bandes et des canaux, augmenter le nombre de ces appareils, une simple modification d'un paramètre ne suffit plus. Il faudrait ici modifier le modèle en lui-même en y ajoutant des stations et peut être en modifiant les transitions entre celles-ci.

Dans la situation où nous nous trouvons, nous sommes réduits à faire une seule chose : fournir comme paramètres obligatoires au modèle tous les paramètres qui ont été mesurés (non compris évidemment ceux destinés à la validation du modèle).

C'est-à-dire

- le temps moyen de service des terminaux, du "CPU", des unités disques, des canaux.
- le nombre moyen de processus transférés de la PWQ vers la TWQ.
- les probabilités de transition à la sortie du "CPU" (4 directions) et le pourcentage de processus se dirigeant vers chaque unité.
- le nombre de processus dans le modèle.
- le nombre moyen de processus dans le "balance set".

De cette manière on effectuera une simulation pour chaque résultat obtenu par le mesureur. Et pour valider ce modèle, on comparera les taux d'occupation des différents serveurs de la simulation avec ceux mesurés.

IV.4.2. L'interactivité de l'outil QNAP

L'outil QNAP tel qu'il est actuellement implémenté à notre institut est très limité quant à l'interactivité. Il permet en fait deux modes principaux :

- Avec le premier, la définition du modèle se trouve en entier dans un fichier et les résultats se trouvent dans un autre fichier. Cependant les paramètres à introduire pour le modèle sont situés à la fin du premier fichier. On peut schématiser cette première situation par la figure IV.9.

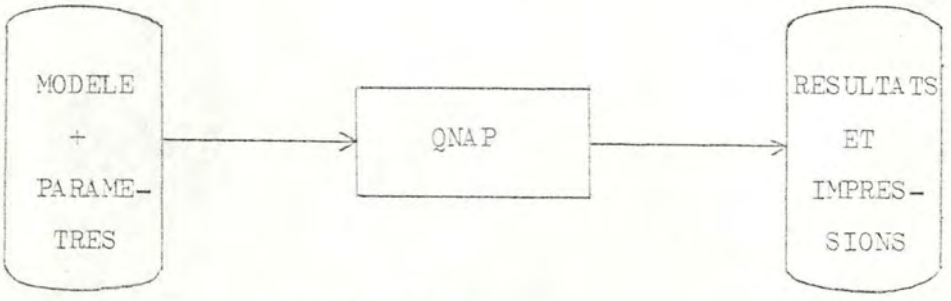


figure IV.9.

- Dans le deuxième mode, tout se passe au contraire aux terminaux. Mais ici, la définition du modèle doit aussi être entrée complètement au terminal. Cette situation est schématisée par la figure IV.10.

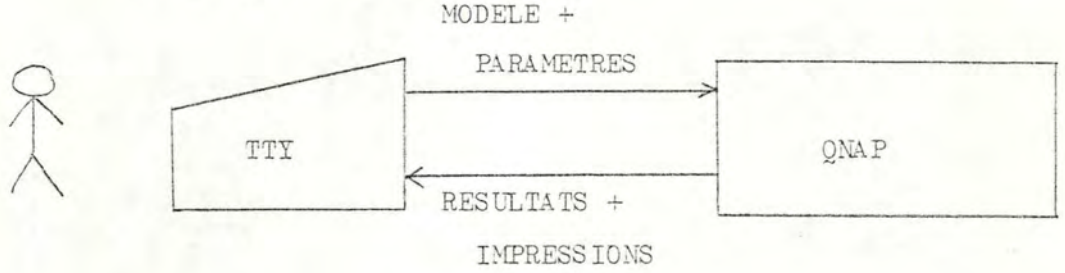


figure IV.10.

Trouvant la deusième solution très contraignante et très fastidieuse, c'est la première que nous avons choisie.

IV.4.3. La longueur de la simulation

Un autre problème qui se pose lorsqu'on résoud des modèles avec la simulation, est de déterminer le temps d'exécution pour obtenir des résultats assez précis. La première chose à respecter, est d'abord d'obtenir un régime stationnaire. Ensuite, QNAP nous permet de calculer pour chaque simulation l'intervalle de confiance des résultats obtenus. Après avoir obtenu la stationnarité, nous déterminerons donc la longueur de la simulation en faisant un compromis entre une durée de simulation assez longue pour que le processus soit stationnaire et pour obtenir un intervalle de confiance assez étroit et une durée de simulation pas trop longue, pour qu'elle ne soit pas trop coûteuse et que le temps d'exécution soit acceptable pour l'utilisateur.

IV.4.4. Les distributions de service

Comme nous l'avons vu dans le chapitre III, nous ne pouvons pas affirmer que les distributions de temps de service sont exponentielles (cfr. III.4.). Cependant les mesures que nous avons effectuées ne nous permettent pas de faire d'autres suppositions sur les distributions : nous ne connaissons que les moyennes et parfois la variance. Sachant que les services ne s'éloignent quand même pas trop de la distribution exponentielle, c'est elle que nous adoptons pour chaque station.

IV.4.5. Le problème des classes de processus

Lorsque nous avons voulu implémenter le modèle, nous avons constaté que la différenciation des processus en classes selon leur file auxiliaire (II.3.) était un gros problème : lorsque nous simulions avec des classes de processus, les résultats obtenus s'écartaient fort de la réalité alors que ce n'était pas le cas si nous n'implémentions pas de classe.

Cela est dû à plusieurs facteurs :

1) Tout d'abord si nous réussissons à mesurer le nombre moyen de processus dans chaque file effectuant des Entrées/Sorties aux terminaux, nous ne pouvons les mesurer dans le "balance set" qu'en les différenciant par groupe de files : "High queues" (files 0 - 1 - 2 et 3) et "Low-queues" (files 4 et 5). Nous ne pouvons donc, pour les processus appartenant au "balance set", qu'estimer approximativement le nombre d'entre eux appartenant à chaque file.

2) Supposons que l'on connaisse exactement le nombre moyen de processus dans chaque file, cela n'est pas pour autant gagné.

Le nombre qu'on connaîtrait alors serait un nombre réel et pas un nombre entier. Or, si l'on doit initialiser un nombre de processus dans chaque file, il faut que ce soit un nombre entier. Dès lors, si l'on arrondit ces nombres réels, la proportion de processus appartenant à chaque file ne sera pas conservée intacte,cela peut avoir de forte conséquences sur le modèle, car dans celui-ci, le temps de service du "CPU" et des terminaux, ainsi que les probabilités de transition sont tous différents selon la classe du processus.

3) Enfin, une dernière raison, mais qui n'est pas la moindre, est le caractère dynamique des files auxiliaires (cfr. I.2.3.). En effet, un processus n'est pas fixé dans une file auxiliaire, il effectue des mouvements ascendants et descendants entre celles-ci. Dès lors pour tenir compte de ce caractère dynamique, nous aurions dû d'abord avoir un mesureur plus complet et ensuite élaborer un modèle plus complexe. Or, dès le départ nous nous étions fixé au nombre moyen de processus et à une répartition statique entre les classes.

Pour toutes ces raisons, l'implémentation des classes a rendu les résultats très imprécis et nous nous en sommes donc tenus à une simulation excluant cette distinction entre classes.

IV.4.6. La programmation du modèle

Pour décrire le modèle, nous devons à chaque serveur du réseau de la figure IV.4. faire correspondre une station dans le langage QNAP.

- les terminaux deviennent une station avec une infinité de serveurs servant les processus selon une discipline FIFO. (NB : Si nous avions tenu compte des classes, le paramètre de la distribution du service aurait été différent pour chaque classe).

- le "CPU" devient un serveur simple servant les clients de sa file d'attente selon une discipline FIFO.
  (NB : Si nous avions tenu compte des différentes classes, ce serveur aurait servi les clients selon une discipline FIFO mais en tenant compte aussi des priorités de chaque classe, le paramètre de distribution de service aurait été différent pour chaque classe, et les probabilités de transition à la sortie de cette station auraient également différé selon les classes).
- A chaque unité disque et à chaque canal correspond un serveur simple de discipline FIFO.

Cependant à cela doivent s'ajouter quelques précisions. Nous avons vu que pour modéliser l'allocation de la mémoire centrale, nous considérions uniquement le nombre de processus présents dans le niveau interne. Un processus peut y entrer, si le nombre de processus s'y trouvant déjà est inférieur au nombre maximum. Quand un processus en sort, il permet à un autre d'y entrer. Cela est implémenté grâce à un sémaphore. Dans le langage QNAP, un sémaphore est une station en elle-même. Elle comporte un compteur qui, s'il est positif désigne le nombre de processus pouvant encore "passer" le sémaphore et s'il est négatif désigne le nombre de clients attendant de "passer" le sémaphore. Elle possède également, comme toute autre station, une file, composée des processus attendant de "passer" le sémaphore.

Cependant, les actions sur ces sémaphores doivent être faites à partir d'autres stations. Il y a donc une station pour les processus voulant "passer" le sémaphore et effectuant l'opération suivante

P (SEM) : - si le compteur du sémaphore est positif le processus "passe" le sémaphore sans problème et le compteur se décrémente d'une unité.

- si le compteur est négatif, ou nul, le processus est mis dans la file d'attente du sémaphore et le compteur est décrémenté d'une unité.

Cette station implémente le serveur représenté par Δ sur la figure IV.4. et traite les processus selon leur priorité.
Il y a donc aussi deux stations représentées par ∇ sur la figure IV.4. et effectuant le service suivant.

V (SEM) : incrémente le compteur du sémaphore d'une unité.

Ces deux stations sont composées de serveurs infinis, n'entraînant donc aucune attente.

Pour implémenter l'allocation de la mémoire centrale, nous avons donc trois stations (Δ et ∇ ∇) plus une station étant le sémaphore lui-même.

Une autre chose très importante est que, après le positionnement sur un cylindre et après le transfert de toutes les demandes concernant ce cylindre de la PWQ vers la TWQ, le disque doit rester occupé jusqu'à ce que toutes ces demandes transférées aient été satisfaites par le canal. Cela est implémenté grâce à un système de flags. Le processus effectuant le positionnement de l'unité transfère toutes les demandes vers la TWQ, crée un processus identique à lui-même, le transfère dans la TWQ, met le flag dans l'état "RESET", et attend qu'il revienne dans l'état "SET". A ce moment, il se tue. C'est le canal qui, quand il a traité le processus qui avait été créé, met le flag dans l'état "SET".

Un autre problème pour les canaux reliés à plusieurs unités (CANAL 0, CANAL 1) est de traiter en un bloc les demandes provenant de chaque unité. Cela est également résolu par un système de flags.

IV.5. RESULTATS FOURNIS PAR LE MODELE

Les résultats produits par les simulations du modèle se trouvent en détail en annexe. Ces simulations sont au nombre de 8 : en effet, à chaque résolution correspond une exécution du programme de mesures sur le système réel. Comme au total, nous avons effectué 8 fois des mesures pendant 1 heure, nous avons obtenus 8 échantillons des paramètres d'entrée au modèle. A partir de ces paramètres, nous avons fait "tourner" notre modèle.

Rappelons que les simulations donnent par station les résultats suivants :

- temps moyen de service : temps moyen pendant lequel les clients (ceux qui ont quitté la station) ont occupé un serveur de la station.
- pourcentage d'utilisation : pourcentage de temps pendant lequel la station a été occupée par au moins un client. Attention : les valeurs données par la simulation sont relatives par rapport à 1 : si l'on veut un pourcentage, il faut multiplier par 100.
- nombre moyen de clients présents dans la station.
- temps moyen de réponse : temps moyen passé par les clients dans la station (à attendre ou à être servi).
- nombre total de clients ayant quitté la station.

IV.5.1. Etat stationnaire

Pour pouvoir obtenir des résultats suffisamment exacts et précis, nous avons dû déterminer la longueur de la simulation. Celle-ci devait être d'une part assez grande pour que le modèle ait atteint un régime stationnaire, mais d'autre part assez courte pour le temps d'exécution "CPU" et l'attente de l'utilisateur. Voici comment nous avons procédé : en observant les différents temps de service des stations, nous avons estimé qu'une longueur de simulation de 1,000,000 unités devrait être adéquate.

Nous avons donc exécuté une simulation (avec les paramètres correspondant à la 7ème exécution du mesureur) durant 1,000,000 unités en prenant soin que celle-ci nous fournisse les probabilités marginales des différentes stations, c'est-à-dire les probabilités qu'il y ait 0, 1, 2, ... n clients présents dans chaque station. Ces probabilités serviraient de critère de comparaison pour établir la stationnarité. Ceci étant fait, nous avons refait la même simulation (avec les mêmes paramètres) mais durant 1,500,000 unités (soit 1,5 x plus longtemps), et de nouveau en demandant dans les résultats les probabilités marginales. Il ne nous restait plus qu'à comparer les probabilités marginales obtenues (ainsi que les autres résultats) des deux simulations et voir si celles-ci avaient changé, auquel cas on aurait pu affirmer que le régime stationnaire n'était pas encore atteint après 1,000,000 unités.

En examinant les résultats de ces 2 simulations (cfr. Annexe H), vous pourrez voir qu'ils ne diffèrent presque pas et donc qu'on peut affirmer que la simulation a déjà atteint sa stationnarité après une durée de 1,000,000 unités. C'est donc cette longueur de simulation que nous avons adoptée pour les 8 simulations.
Voici dans ce tableau la comparaison de quelques résultats fournis par les 2 simulations.

long. simulation / résultats de la simulation	1,000,000	1,500,000
% utilisation "CPU"	47,73	46,96
% utilisation PS:1	29,15	28,58
% utilisation CANAL 0	20,26	19,72
Nbre moyen clients TTY	24,92	24,97
Nbre moyen clients CPU	0,8741	0,8473
Temps moyen service TTY	2167	2198
Temps moyen service CPU	11,02	11,05
Temps moyen service CANAL 0	16,18	16,07

Comparons maintenant les probabilités marginales de quelques stations :

Station "CPU"

durée 1,000,000 → Nbre de clients	0	1	2	3	4	5
Probabilité	0,523	0,259	0,121	0,057	0,025	0,009

durée 1,500,000 → Nbre de clients	0	1	2	3	4	5
Probabilité	0,53	0,257	0,119	0,055	0,024	0,008

Station PS:1

durée 1,000,000 → Nbre de clients	0	1	2	3	4	5
Probabilité	0,708	0,248	0,038	0,006	0,001	0,000

durée 1,500,000 → Nbre de clients	0	1	2	3	4	5
Probabilité	0,714	0,243	0,037	0,005	0,001	0,000

Station CANAL 1

durée 1,000,000 → Nbre de clients	0	1	2	3	4	5
Probabilité	0,871	0,114	0,013	0,002	0,000	0,000

durée 1,500,000 → Nbre de clients	0	1	2	3	4	5
Probabilité	0,875	0,111	0,013	0,002	0,000	0,000

Nous constatons que les résultats diffèrent très peu entre les deux simulations. Nous pouvons donc affirmer que l'état stationnaire est déjà atteint pour une durée de 1,000,000 unités.

IV.5.2. Intervalle de confiance

QNAP nous permet de demander que les différentes simulations nous fournissent un intervalle de confiance pour les estimations des résultats. Nous avons demandé cet intervalle pour toutes les stations du modèle (excepté WAITMEM, LIBTTY et LIBFIL). Ces intervalles sont établis à un niveau de confiance de 95 %. Chaque intervalle de confiance doit être interprété comme suit : il y a 95 % de chances que la valeur exacte, celle correspondant à l'état stationnaire du modèle, soit dans cet intervalle.

Mais comment QNAP construit-il ces intervalles ?
Le principe de base est de diviser la durée de la simulation en plusieurs blocs successifs, de sorte que le comportement du modèle dans ces intervalles soit statistiquement équivalent et indépendant. Il y aura donc autant de sous-simulations indépendantes qu'il y a de blocs. On calcule alors la moyenne des observations dans chaque sous-simulation ; on obtient ainsi un échantillon de moyennes indépendantes. La variance de cet échantillon des moyennes se calcule alors sans problème et permet d'établir l'intervalle de confiance cherché.

Diviser la durée de la simulation en des périodes de taille fixe produit des blocs qui satisfont l'hypothèse d'indépendance si la taille de ces blocs est suffisamment large. Cette méthode est la méthode par défaut (celle que nous avons utilisée) dans QNAP (l'exécution de la simulation est divisée en 100 périodes de longueur fixe). Il faudra donc que la durée de la simulation soit assez grande, d'autant plus que les intervalles de confiance seront d'autant plus précis et étroits que la simulation dure longtemps.
Une règle empirique donnée dans le manuel de référence de QNAP est de dire que l'on devrait avoir au moins 1,000 services exécutés dans une station pour que les résultats soient significatifs.

Examinons quelques intervalles de confiance ainsi que le nombre de services associés à quelques stations. Nous les avons puisés dans les résultats de la 1ère simulation (dont les paramètres proviennent de la 1ère exécution du mesureur).

Noms	tps moyen service	% utilisation	Nbre moyen clients	tps moyen réponse	Nbre services
"CPU"	8,633	0,4494	0,778	14,77	52669
+ / -	0,06871	0,009296	0,03916	0,5361	
PS:1	32,67	0,3453	0,4257	40,28	10569
+ / -	0,4421	0,01165	0,01919	0,8841	
US	41,63	0,05779	0,05932	42,74	1388
+ / -	1,697	0,003963	0,004144	1,663	
MONT	39,68	0,0006694	0,0006694	39,38	17
+ / -	9,65	0,0003552	0,0003552	9,65	

On constate donc que les intervalles sont suffisamment précis, même pour l'unité US où le nombre de passages est de 1388. Par contre, les intervalles concernant le disque montable sont beaucoup trop larges (relativement à l'ordre de grandeur de la valeur centrale), ce qui est normal vu le nombre très faible de services dans cette station (17). Si l'on regarde les paramètres d'entrée, on comprendra pourquoi : le disque montable n'a qu'un pourcentage d'importance de 0,06 par rapport aux autres unités.

IV.5.3. Validité du modèle

Pour vérifier que le modèle est conforme à la réalité, nous allons comparer les pourcentages d'utilisation des différents serveurs, d'une part calculés sur le système réel lors des 8 exécutions du mesureur (1) et d'autre part fournis par les simulations (2).
Nous n'avons pas envisagé les pourcentages d'utilisation des terminaux, vu qu'ils n'ont pas été calculés lors des mesures et que dans notre modèle, ils sont représentés par une unique station à une infinité de serveurs.

De plus, les pourcentages d'utilisation des unités (PS:1, PS:2, PS:3, US, MONT) sont nécessairement plus grands dans les résultats du modèle, vu que le service de ces unités comprend non seulement le temps de positionnement mais aussi des temps d'attente de positionnement de 1 ou 2 flags (attente de fin de transfert de toutes les demandes de l'autre unité vers la TWQ (*) et attente de fin du transfert par le canal de toutes les demandes transférées d'un coup de la PWQ vers la TWQ).

Voici un tableau de comparaison des pourcentages d'utilisation pour les 8 exécutions du mesureur et du modèle.

mesureur (1) et modèle (2) / serveurs		I	II	III	IV	V	VI	VII	VIII
"CPU"	(1)	44,46	52,45	45,95	43,51	42,54	46,53	46,7	34,08
	(2)	44,94	55,48	48,48	42,95	42,59	47,78	47,73	34,71
CANAL 0	(1)	20,47	32,38	17,89	20,59	17,53	24,55	19,02	14,58
	(2)	20,43	33,92	19,65	20,92	17,6	25,87	20,26	14,93
CANAL 1	(1)	18,01	18,25	14,78	12,41	16,72	18,27	12,44	12,5
	(2)	18,76	19,22	16,08	12,59	17	18,71	12,87	13,17
CANAL 2	(1)	17,42	18,67	15,34	14,47	14,42	19,27	14,19	12,29
	(2)	17,15	19,97	16,27	14,51	14,12	20,02	15,25	12,65
CANAL 3	(1)	37,22	6,14	5,6	7,43	8,92	11,98	6,7	4,83
	(2)	38,58	6,521	5,889	7,776	9,423	12,93	8,071	12,6

On peut remarquer que les valeurs ne diffèrent vraiment pas beaucoup deux à deux, ce qui amènerait à croire que notre modèle est assez proche de la réalité.

(*) uniquement dans le cas de 2 unités associées à un même canal.

IV.5.4. Analyse des autres résultats

IV.5.4.1. Temps moyen de service

Nous présentons dans le tableau suivant le temps de service des différentes stations, d'une part ceux calculés par le mesureur (1) et d'autre part ceux fournis par le modèle (2). De nouveau, nous ne tiendrons pas compte des différentes unités (PS:1, PS:2, PS:3, US, MONT), vu que leur temps de service calculé par le mesureur correspond au temps moyen de positionnement alors que dans le modèle, il faut ajouter à ce temps les différents temps d'attente devant les flags compris dans le service des unités.

Nous avons estimé qu'il était suffisant d'établir les comparaisons pour les 4 premières simulations, car ces comparaisons ont été établies, non point pour valider le modèle (il est en effet logique que les moyennes soient sensiblement égales, vu que celles calculées par le mesureur servent de paramètres d'entrée pour le service des différentes stations du modèle), mais pour s'assurer du bon fonctionnement des générateurs de nombres aléatoires du simulateur.

Nom	Mesureur (1) et Modèle (2)	I	II	III	IV
TTY	(1)	1966,555	1826,5049	2055,9435	2024,693
	(2)	1994	1849	2096	2070
	+/-	34,78	33,02	40,81	34,82
"CPU"	(1)	8,4382	9,3375	11,4757	9,423
	(2)	8,533	9,475	11,63	9,475
	+/-	0,068	0,078	0,1066	0,08951
CANAL 0	(1)	17	16	16	16
	(2)	17,09	16,15	16,19	16,04
	+/-	0,3035	0,2456	0,2703	0,2851
CANAL 1	(1)	17	17	17	18
	(2)	16,97	17,08	17,03	18,04
	+/-	0,3637	0,3019	0,3248	0,3885
CANAL 2	(1)	18	18	18	16
	(2)	17,88	18,06	18,06	16,17
	+/-	0,3687	0,3001	0,3570	0,3522
CANAL 3	(1)	54	183	199	50
	(2)	54,18	174,4	179	50,62
	+/-	1,292	16	20,37	2,785

Nous remarquons que les valeurs obtenues de chaque côté sont sensiblement égales et qu'en général l'intervalle de confiance fourni par le modèle recouvre la valeur mesurée.

IV.5.4.2. Temps moyens de réponse

Nous pouvons formuler deux remarques :

- d'une part, les temps moyens de réponse des stations à infinité de serveurs sont égaux aux temps moyens de service, ce qui est tout à fait normal vu que le temps d'attente est alors nul.

ex. : Simulation II

Nom	temps moyen de service	temps moyen de réponse
TTY	1849	1849

Ces 2 temps sont aussi égaux dans le cas d'une station où le nombre de services durant la simulation a été très petit, ce qui implique qu'il n'y a pas eu d'attente.

ex.

	Nom	temps moyen de service	temps moyen de réponse	Nombre services
Simulation I	MONT	39,38	39,38	17
Simulation II	MONT	27,8	27,8	3
Simulation IV	MONT	45,96	45,96	19

- d'autre part, on peut remarquer que pour le "CPU" le temps moyen de réponse est assez important relativement au temps moyen de service, ce qui peut s'expliquer par le fait que le "CPU" est la station à plus fort pourcentage d'utilisation et que le nombre moyen de clients présents dans cette station est aussi le plus grand (excepté la station TTY).

CPU	% utilisation	Nombre moyen clients	Temps moyen de service	Temps moyen de réponse
Simulation I	44,94	0,778	8,533	14,77
Simulation II	55,48	1,155	9,475	19,72
Simulation III	48,48	0,8687	11,63	20,84
Simulation IV	42,95	0,7265	9,475	16,03
Simulation V	42,59	0,7356	11,57	19,98
Simulation VI	47,78	0,8821	8,608	15,89
Simulation VII	47,73	0,8741	11,02	20,18
Simulation VIII	34,71	0,5288	9,579	14,59

C O N C L U S I O N S

Dans ce travail, nous sommes maintenant arrivés à un point de développement où nous disposons d'un outil permettant d'évaluer les performances du DECSYSTEM-20 des Facultés Universitaires Notre Dame de Namur. Mais cet outil n'est pas parfait.

Dans un premier temps, nous avons réalisé un mesureur, grâce auquel nous pouvons effectuer des mesures d'une heure sur le système réel. A cause de la compréhension approfondie du système qu'elle nécessitait et le manque de documentation précise concernant certains détails du système, c'est l'élaboration de ce mesureur qui a constitué la partie la plus importante et la plus lonque de notre travail. Quand cette première étape de la réalisation de notre outil fut terminée, nous nous trouvions à un moment où le système n'était pas utilisé de manière normale ; il était très peu utilisé. Nous n'avons donc pu effectuer de mesures très "parlantes".

Si nous devions formuler une critique vis-à-vis de ce mesureur, c'est que d'une part, il ne peut déterminer exactement le nombre de processus dans chaque file auxiliaire et d'autre part, il ne mesure que les moyennes (et les variances) des temps de service des terminaux, processeur, unités disques et canaux et ne permet aucune conclusion quant aux distributions de ces temps.

Dans un second temps, nous nous sommes servis des mesures ainsi effectuées sur le système réel, pour les fournir comme paramètre d'entrée à un modèle de files d'attente schématisant le système. La résolution de ce modèle a été limitée par deux choses : La première est due au fait que les mesures n'ont pu être réalisées pour toutes les charges possibles du système mais uniquement sur un système très peu utilisé. La deuxième est la limitation imposée par le langage de résolution de modèles dont nous disposions (QNAP).

Ce modèle peut être sujet à des modifications dans ses paramètres et dans sa définition pour permettre à l'utilisateur d'une part de l'adapter aux changements qu'il pourrait envisager sur le système, et d'autre part de tester la valeur de ces changements. La critique essentielle vis-à-vis de ce modèle est qu'il n'envisage pas la classification dynamique des processus dans les classes auxiliaires, telle qu'elle se passe dans le système réel.

Ce mémoire nous a permis d'étudier, dans certains de ses détails, un système d'exploitation particulier et de découvrir ses mécanismes. Nous avons également eu l'occasion de savoir ce que signifie réellement la prise de mesures de performances et l'élaboration d'un modèle d'un système, ce dont nous n'avions aucune idée. Par là, nous avons pu découvrir les limites d'un outil tel que QNAP.

S'il nous fallait donner suite à ce mémoire, la première chose à faire serait d'exécuter le mesureur pour les différentes charges possibles sur le système et d'adapter le modèle aux conclusions tirées des mesures obtenues. Ensuite, il faudrait perfectionner ce mesureur pour étudier les distributions de temps de service. Le modèle quant à lui pourrait être adapté pour inclure la classification dynamique des processus dans les files auxiliaires, le modèle ainsi obtenu serait plus proche encore de la réalité.

Bibliographie

[ADVE] J.-P. Adans et J. Vercheval
"Mesure de performances de UNIX sous une charge universitaire"
Mémoire de l'institut d'informatique FNDP 80-81

[AGRA] Ashak K. Agrawala, Satish K. Tripthi, Marc Abrams, K.K. Ramakrishman, Mukesh Singhal, Song H. Son
"STEP-1 : A user friendly Performance Analysis Tool"
INRIA (Institut de Recherche en Informatique et en Automatique)
Colloque international sur la modélisation et les outils d'analyse de performance
(Edition Provisoire) Mai 1984

[BECK] Bruno Beck
"Mesures de performances d'une liaison entre ordinateurs"
Mémoire ULB 81-82

[BLUM] Alvin Blum, Lorenzo Donatiello, Philip Heidelberger, Stephen S. Lavenberg, Edward A. Mac Nair
"Expériments with Décomposition of extended Queuing Network Models"
INRIA, colloque international sur la modélisation et les outils d'analyse de performance - Mai 1984

[BRAN] Alexandre Brandwajn
"Issues in mainframe system modelling - Lesson from model developpement at Andahl"
INRIA, Colloque international sur la modélisation et les outils d'analyse de performance - Mai 1984

[BUZ] Jeffrey P. Buzen
"A queuing network model of MVS"
Computing Surveys - Vol 10 - n° 3 - Sept 78

[CHSA] K. Mani Chandy et Charles H. Sauer
"Approximate methods for analysing queuing network models of computing systems"
Computing Surveys - Vol 10 - n° 3 - Sept 78

[CORN] Dirk Cornil
"Mesures de performances de UNIX à partir d'un modèle mathématique"
Mémoire de l'institut d'informatique FNDP 80-81

[DAWI] Michel Dawirs
"Ordonnancement dans le système DEC-2050"
Travail de fin d'études de l'université de Liège, 1980-1981

[DEBU] Peter J. Denning et Jeffrey P. Buzen
"The operational analysis of queuing network models"
Computing Surveys - Vol 10 - n° 3 - Sept 78

[DIG1] Digital Equipement Corporation
"DEC SYSTEM-20 Monitor Flowcharts" 1978

[DIG2] Digital Equipement Corporation
"DEC SYSTEM-20 Monitor Structure" 1977

[DIG3] Digital Equipement Corporation
"Monitor calls - Reference Manuel" 1980

[DIG4] Digital Equipement Corporation
"TOPS-20 Monitor Internals, Student Guide" 1978

[DIG5] Digital Equipement Corporation
"TOPS-20 Monitor Structures, Student Guide" 1983

[DIG6] Digital Equipement Corporation
"TOPS-20 Monitor Tables" 1978

[FICH] Jean Fichefet
"Cours de Méthodes de Simulation" FNDP

[GOR] Ralph E. Gorin
"Introduction to DECSYSTEM-20/Assembly Language Programming"
Digital Press 1981

[NOIC] Mme Noirhomme-Fraiture
"Cours de système d'exploitation : Modèles et performances"
FNDP

[NOIT] Mme Noirhomme-Fraiture
"Modélisation d'un système d'exploitation : réseaux fermés de files d'attente"
Tutorial Paper XII 1982

[QNAP] QNAP2 : Queuing Network Analysis Package
"Référence Manuel"
CII Honeywell Bull and INRIA, 1982

[UER] René Verhaegen
"Gestion des processus sur DEC-2050"
Institut d'informatique FNDP - 1979

[XIN] Yan Xin
Travail de fin d'études de l'université de Liège,
1981-1982

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX - NAMUR

INSTITUT D'INFORMATIQUE

Mesures de performances : Construction d'un outil de mesures et modélisation du DEC-20

ANNEXES

Christine RABEUX

Gaëtan PLUQUET

Promoteur : Mme Noirhomme - Fraiture

Mémoire présenté en vue
de l'obtention du grade de
LICENCIE ET MAITRISE EN INFORMATIQUE

ANNEE ACADEMIQUE 1983 - 1984

Mémoire de Christine RABEUX
Gaëtan HUGUET

"Mesures de performances :
Construction d'un outil de mesures et
modélisation du DEC-20."

ANNEE 1983-1984

Nous nous excusons auprès des lecteurs de ce mémoire pour les différentes erreurs de dactylographie qu'ils pourraient rencontrer. Certains problèmes ne nous ont pas permis de relire assez en profondeur ce volume avant sa duplication; aussi, en vue de faciliter la compréhension du lecteur, avons nous dressé une liste des erreurs les plus importantes. Dans celle-ci, nous reprenons chaque fois une partie de la phrase où l'erreur se situe et soulignons le ou les mots erronés tels qu'ils doivent être lus.

p. 6. .... et de mémoire virtuelle avec pagination.

p. 10. (I.1.4.) - Courant : le processus occupe le processeur....

p. 18. (2° §) .... le faire passer dans l'état Prêt-Actif. Un processus en attente est donc un processus dans l'état Bloqué-Actif, Bloqué-Inactif ou Prêt-Inactif .....

p. 19. (3.) crédit + reste du "quantum" > 3000 ms .....

p. 24. (1° §) .... décrire la manière dont un processus ....
(dern.§) .... : le nombre maximum de processus lui appartenant, donc ....

p. 26. ((1)) .... disponible pour les processus utilisateurs ....
.... et volontairement du balance set lorsqu'il effectue ....

(1)) (Cette méthode est connue dans la ....
.... "Round Robin Method").

p. 28. (dern.§) .... scheduler qui décide quel processus à exécuter.

p. 29. (I.5.) ...., imprimantes, ... qui elles sont longues.

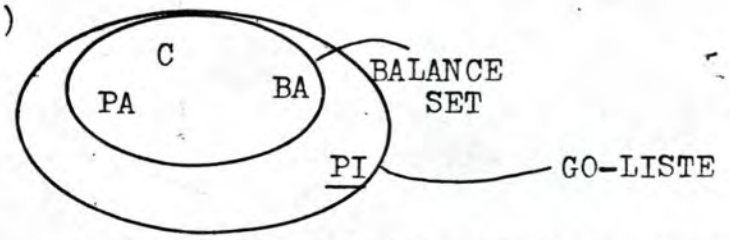
p. 30. (I.5.2.) .... préférence aux demandes de lecture. En favorisant la lecture, on tend à réduire le temps d'attente des processus dû au swapping car un processus attend généralement pour une lecture tandis que les écritures sont plutôt générées par la "Local Garbage Collection".

p. 31. .... supérieur pour lequel il existe une demande de lecture.
.... supérieur pour lequel il existe une demande d'écriture.
.... inférieur pour lequel il existe une demande de lecture.
.... inférieur pour lequel il existe une demande d'écriture.

p. 33. (I.6.1.) .... des valeurs s'étendant de 1 à 20, ...

p. 34. (2° §) .... et non chaque processus pris ....

p. 38. (fig.I.5.)



- Les processus Bloqués-Inactifs ne se trouvent pas dans la GO-LISTE, ...

p. 39. (fig.I.6.) La transition PA ⟵ BA doit être orientée dans ce sens.
La transition BA ⟶ BI ne doit pas exister dans cette figure.
Création d'un processus.

p. 49. (1° §) .... centrale (file devant Δ)....

p. 51. ((3)Rem.) .... selon les niveaux de priorité des processus,...

p. 59. (1° §) .... du bras de lecture) (cfr.I.5.2.).

p. 77. (2° §) .... cumuler sur "i" les "λi" et "$\tilde{\mu}$i").

p. 87. (3° §) .... dans les listes d'attente pour entrées/sorties sur terminal .....

p. 92. .... plus précis vu que la taille de l'échantillon.

p. 117. (IV.2.5.) 4) Quand le sous-modèle contient plusieurs chaînes fermées.

p. 124. (2° §) .... théorie de la décomposabilité presque complète (cfr.IV.2.6.): ....
.... (.... l'autre la synchronisation des disques et canaux).

p. 129. (2° §) .... décompositions précédentes, le troisième niveau d'agrégation respecte la théorie de la décomposabilité presque complète, par contre, le deuxième ne la respecte pas.

p. 156. [VER] René Verhaegen

ANNEXES

ANNEXE A : Principales tables du moniteur

ANNEXE B : Tables accessibles par GETAB

ANNEXE C : Modules du moniteur utilisés

ANNEXE D : QNAP

ANNEXE E : Le programme du mesureur

ANNEXE F : Les mesures obtenues

ANNEXE G : Le modèle programmé en QNAP

ANNEXE H : Les résultats du modèle avec les probabilités marginales (prouvant la stationnarité)

ANNEXE I : Les résultats du modèle

ANNEXE A

PRINCIPALES TABLES DU MONITEUR

Cette annexe contient une brève description des principales tables du moniteur.

Commençons d'abord par celles que nous avons employées

CDB : Channel Data Block

Cette table, une par canal, contient des instructions et des données pour le canal associé : des informations sur le canal, des pointeurs vers les unités (UDB) du canal, des informations sur l'unité courante, le "compteur de fiabilité" (voir étude du système), des instructions de contrôle pour l'interruption du canal ...

FKQ : Fork Run Queue Table

Il existe deux tables FKQ. Elles décrivent les files auxiliaires des processus.
La première (FKQ1), contient pour chaque processus, le reste de quantum qu'il peut encore exécuter dans la file auxiliaire où il se trouve.
La deuxième (FKQ2), contient pour chaque processus, le numéro de la file auxiliaire où il se trouve, et l'adresse de la liste où il se trouve (TTYLIST, GOLST ...) ou l'index de la table BALSET si le processus se trouve en mémoire centrale.

IORB : Input/Output Request Block

Quand une demande d'entrée/sortie pour diques et bandes se produit, on construit un bloc pour cette demande. Si la demande est pour une bande, ce bloc est long, pour un disque, il est généralement court, mais, quelques fois, il arrive qu'il soit long.
Il contient notamment un "status", un pointeur vers l'IORB suivant, l'adresse de l'unité physique (si nécessaire) ...

KDB : Kontroller Data Block

Uniquement dans le cas de bandes magnétiques, un contrôleur s'intercale entre le canal (CDB) et les unités (UDB).
Ce bloc, un par contrôleur contient des informations sur le contrôleur associé et notamment des pointeurs vers ses unités.

SDB : Structure Data Block

Une structure est la manière dont les informations sont mémorisées. Une structure peut s'étendre sur plusieurs unités et il existe plusieurs types de structures. La structure PS est la "Public structure" et elle s'étend sur trois unités (PS1, PS2, PS3).
Le bloc SDB contient des informations sur les unités de la structure, sur la "master directory" (ex.: roat-directory) et sur la structure elle-même.

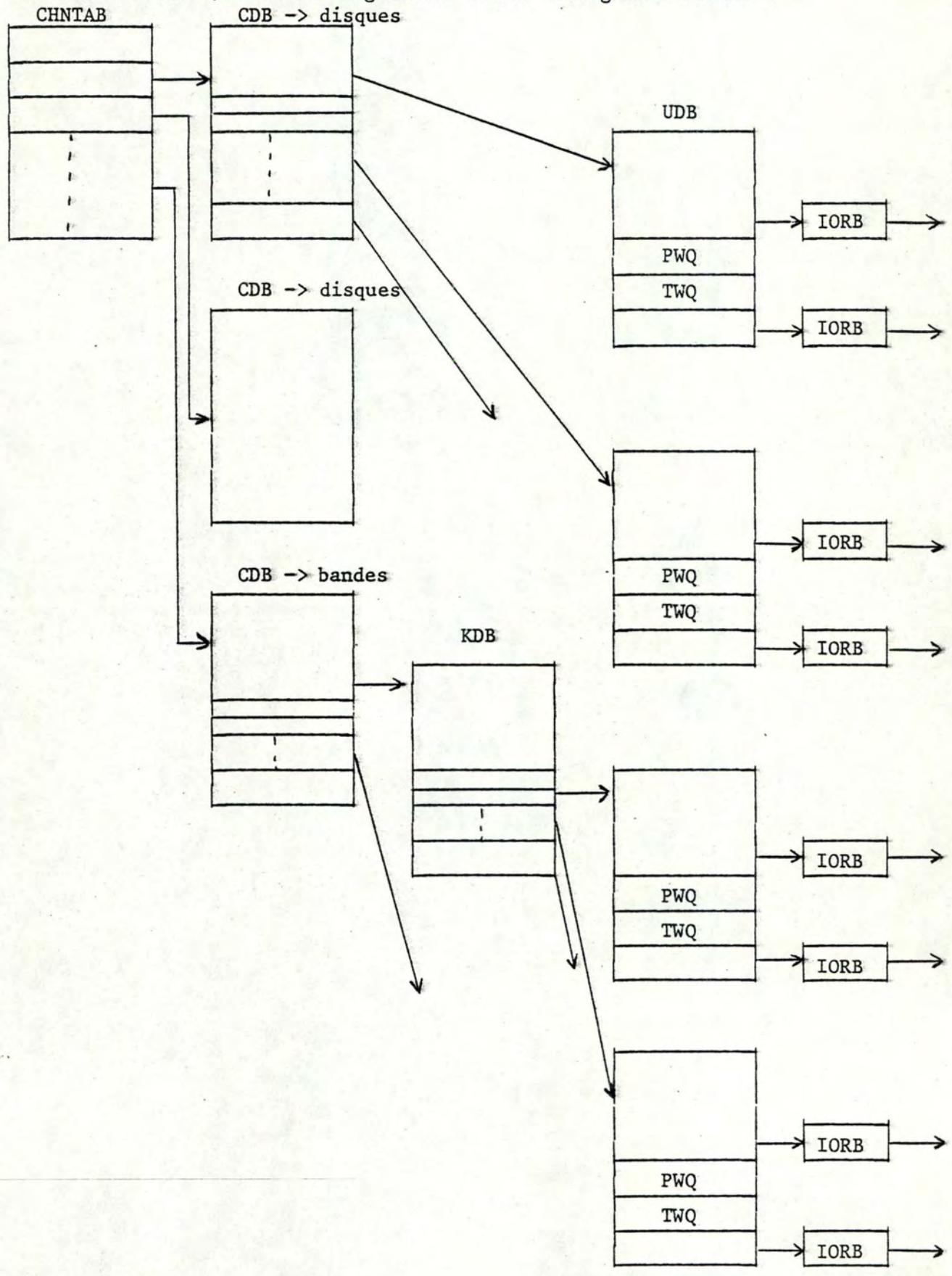
STR TAB : Structure Data Block

Cette table, indexée par le numéro de la structure, contient un pointeur vers chaque SDB du système.

UDB : Unit Data Block

Cette table, une par unité, contient des informations sur l'activité de l'unité, telles que des pointeurs vers le CDB, vers le SDB, vers le KDB (si il existe), vers les files d'attente pour le positionnement (PWQ) et pour le transfert (TWQ), telles que la position courante du bras, le processus qui possède l'unité ...

Les blocs de données pour les canaux, les contrôleurs, les unités et pour les demandes d'entrée/sorties s'organisent selon la figure suivante :



La table CHNTAB (décrite ci-après) contient l'adresse du CDB de chaque canal. Dans le cas des disques, le CDB d'un canal contient l'adresse de tous les UDB's des unités ratachées à ce canal. Tandis que pour les bandes, le CDB contient l'adresse de tous les KDB's attachés et le KDB contient l'adresse de tous les UDB's attachés. A chaque unité sont associées deux files d'IORB's : celle pour le positionnement (PWQ) et celle pour le transfert (TWQ). L'UDB de chaque unité contient un pointeur vers le début de chaque file et chaque IORB pointe vers l'IORB suivant dans la file.

Décrivons maintenant quelques autres tables importantes du système.

BALSET : Balance Set Table

Cette table contient tous les processus ayant leur "working set" en mémoire centrale (voir chapitre I : étude du système). Et pour chaque processus, un status, indiquant si ce processus attend pour une E/S courte (disque ou bande) ou non...
Cette table n'est pas ordonnée.

BSPT : Balance Set Process Table

Cette table est parallèle à la table BALSET. Elle permet d'ordonner BALSET grâce à deux séries de pointeurs. A chaque processus de "balset" sont associés un pointeur vers le processus précédent et un pointeur vers le processus suivant. Les pointeurs vers le début et la fin de la liste sont gardés dans la partie résidente du moniteur.

BSQ : Balance Set Quantum Table

Cette table contient pour chaque processus de "balset", son "balance set quantum".

CHNTAB : Channel Table

Cette table (indexée par le numéro de canal) contient l'adresse de tous les CDB's.

DEVCHR : Device Characteristic's Table

Cette table (indexée par le numéro de "service") contient les caractéristiques, le type et les modes de travail de chaque "device".

FKJOB : Fork Job Table

Cette table (indexée par le numéro du processus) contient pour chaque processus, le numéro de son "job" et l'adresse du JSB (décrit ci-après) de son "job".

FKPT : Fork List Pointer Table

Cette table (indexée par le numéro du processus) contient pour chaque processus du système un pointeur. Il sert à enchainer les processus de chaque liste de processus du système (ex. : TTILST, CLKLST, GOLST,...).

JOBPT : Job Process Table

Cette table (indexée par le numéro de "job"), contient pour chaque "job" du système, le numéro de terminal de contrôle de ce "job" ("-1" si le "job" est détaché) et l'index du processus de niveau supérieur de ce "job".

JOBRT : Job Runtime Table

Cette table (indexée par le numéro de "job"), contient pour chaque "job", le temps total pendant lequel il s'est exécuté.

JSB : Job Storage Block

Un tel bloc existe pour chaque "job" du système. Il contient des informations telles que : la structure du bloc, le numéro du terminal de contrôle, une table de correspondance entre les adresses initiales et les adresses en mémoire centrale des pages du "job", et beaucoup d'autres informations sur les pages du "job".

PSB : Process Storage Block

Ce bloc est semblable au précédent, mais pour chaque processus du système. Il contient des informations telles que : le PC et les AC's du processus quand il ne s'exécute pas, la table de correspondance entre pages mutuelles et pages en M.C et beaucoup d'autres informations sur les pages du processus.

A N N E X E B

PRINCIPALES TABLES ACCESSIBLES PAR GETAB

Dans cette annexe, nous nous contenterons d'exposer les tables que nous avons jugées les plus importantes ainsi que celles que nous avons utilisées pour nos mesures. Pour plus d'informations, il faudra se référer au "Monitor Calls Reference Manuel".

L'appel GETAB accepte comme paramètre le numéro d'identification d'une table fourni par l'appel SYSGT ou obtenu dans le fichier MONSYM et il permet d'entrer dans cette table pour y puiser l'information contenue.

Nous donnerons pour chaque table retenue, le nom de la table, l'indice pour accéder à un élément de cette table (si il en existe un) et une explication du contenu de cette table.

Nom	Indice	Contenu
APRID		Numéro de série du processus
BLDTD		Date et heure de la génération du système
DBUGSW		Informations sur le "debugging"
DEVCHR	DEV#	Caractéristiques d'appareil ("device")
DEVNAM	DEV#	Nom du "device" en SIXBIT incluant le n° d'unité, p. ex., MTA 3.
DEVUNT	DEV#	MG : numéro du job auquel le "device" est assigné, ou -1 si le "device" n'est pas assigné, ou -2 si il est réservé pour l"allocateur de device" MD : numéro d'unité, ou -1 si le "device" n'a pas d'unités (p. ex. : DSK)
DRMERR		informations sur les erreurs sur tambours ("drum")
DSKERR		informations sur les erreurs sur disque
DWNTIM		informations sur le moment d'arrêt du système
HQLAV		"High Queue Load Averages" : nombre moyen de processus présents dans la GOLISTE et les files auxiliaires 0, 1, 2 ou 3
IMPLT1 IMPLT2 IMPLT3 IMPLT4		informations sur l'utilisation du réseau ARPANET (numéro de connection interne, adresses des buffers, etc...)
JBONT	JOB#	"job" propriétaire pour les "jobs" créés par CRJOB
JOBNAM	JOB#	MG : réservé pour DEC MD : indice dans les tables du système pour le programme - système utilisé par ce "job"
JOBPNM	JOB#	Nom en SIXBIT du programme s'exécutant dans ce "job"
JOBRT	JOB#	temps "CPU" utilisé par le "job"

Nom	Indice	Contenu
JOBTTY	job#	MG : numéro de ligne du terminal qui contrôle le "job" ou -1 s'il n'y en a pas ("job" détaché)
LOGDES		information sur le "logging"
LQLAV		"Low Queue Load Averages" : nombre moyen de processus présents dans la GOLISTE et les files auxiliaires 4 ou 5
NETHST NETAWD NETBAL NETBTC NETBUF NETFSK NETLSK		différentes informations à propos de chaque connection interne (buffers, nombre de bits envoyés ou reçus, etc ...)
NETRDY		"status table" opérationnelle d'ARPANET
NCPGS		Table à un mot contenant le nombre de pages physiques de mémoire ferrite ("core") disponibles dans le système pour les utilisateurs
NSWPGS		pages pour le "swapping" par défaut
PTYPAR		informations sur les pseudo-TTY (nombre et n° TTY du premier)
QTIMES	0 → n	Temps d'exécution accumulé par les "jobs" dans les n files du "scheduler"
SNAMES		Nom SIXBIT du programme système
SNBLKS		Nombre d'échantillons dans le calcul de l'intégrale de la taille du working-set
SPFLTS		Nombre total de défauts de page du programme système
SSIZE		Intégrale dans le temps de la taille du "working set"
STIMES		Temps total d'exécution du programme système

Nom	Indice	Contenu
SYMTAB		Nom en SIXBIT de toutes les tables GETAB
SYSTAT		Table très importante qui fournit les statistiques du moniteur. Il est à noter que cette table peut être modifiée ou augmentée dans le cas où des routines de mesure seraient ajoutées au système
	0	Temps inutilisé : aucun "job" n'est exécutable ("runnable")
	1	Temps d'attente avec un ou plusieurs "jobs runnables" (attente de transfert de pages vers la mémoire centrale)
	2	Temps passé dans le "scheduler"
	3	Temps passé pour exécuter le "trapping" du "pager"
	12	Intégrale dans le temps du nombre de processus dans le "balance set"
	13	Intégrale dans le temps du nombre de processus "runnable"
	32	Temps utilisateur
	33	Intégrale dans le temps du nombre de processus dans "High Queue" (files 0, 1, 2 et 3)
	34	Intégrale dans le temps du nombre de processus dans "Low Queue" (files 4 et 5)
	47	Intégrale du nombre de "working sets" en mémoire

Nom	Indice	Contenu
SYSVER		"String" ASCIZ identifiant le nom du système, la version et la date
TICKPS		table d'un mot contenant le nombre de "clock ticks" par seconde
TTYJOB	line#	MG : numéro positif du job contrôle par le terminal, ou -1 si la ligne n'est pas assignée, ou -2 si la ligne est couramment assignée ou le numéro de "job" auquel elle est assignée MD : -1 si aucun processus n'est en attente pour une entrée à ce terminal, sinon une autre valeur que -1

ANNEXE C

MODULES DU MONITEUR UTILISES

1. Gestion et allocation des processus et de la mémoire

PAGEM

Ce module gère les pages du système. Il contient le code des routines de gestion de la mémoire, de gestion des "swapping", la gestion des défauts de pages ...

SCHED

Ce module gère les processus, et l'accès de ceux-ci à la mémoire centrale. Il contient les routines de changement de contexte, le "scheduler", les routines d'initialisation des processus, les routines de désactivation des processus ...

2. Gestion des unités

PHYH2

Ce module gère les contrôleurs : initialisation, terminaison et recouvrement d'erreurs. Quand une interruption survient, il effectue l'analyse principale et appelle la routine pour l'unité appropriée.

PHYM2

Ce module contient le code pour la gestion des bandes magnétiques.

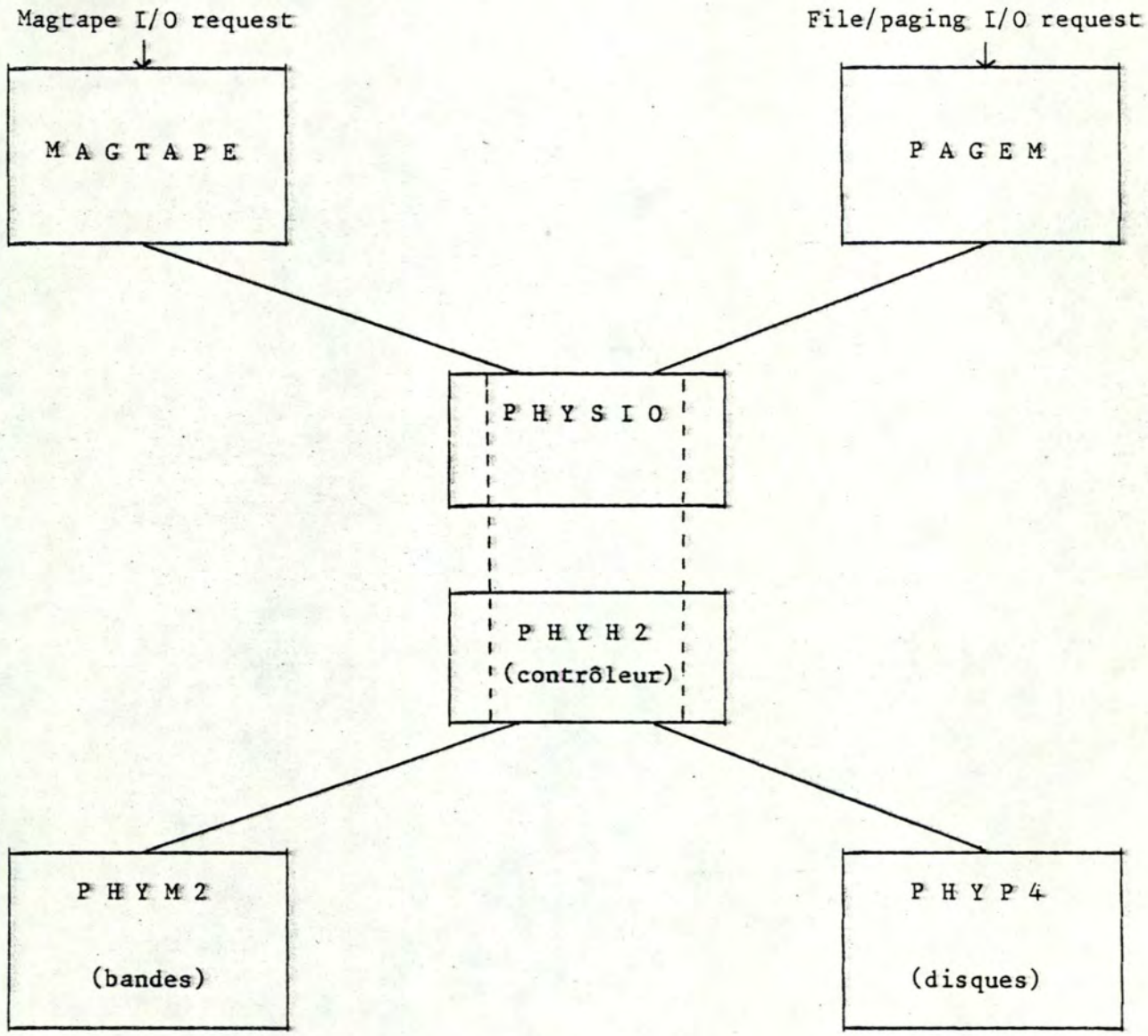
PHYP4

Ce module contient le code pour la gestion des disques.
Il traite, le positionnement du bras, le transfert et le recouvrement d'erreurs pour les disques.

PHYSIO

Ce module gère les Entrées/Sorties pour les canaux et les unités disques et bandes. Il met dans la file correcte les demandes d'E/S, choisi la "meilleure" demande pour le positionnement et demande l'E/S.
Il traite les interruptions venant de PHYH2.

Ces modules pour la gestion des Entrées/Sorties sur disques et bandes s'organisent selon le schéma suivant.



Les demandes d'E/S sont soumises à PHYSIO via les modules PAGEM et MAGTAP. PHYSIO traite les E/S physiques et passe la main aux modules PHYH2, PHYM2, PHYP4 pour effectuer leurs fonctions.

3. Gestion de la Base de données

PHYPAR

Ce module est en fait un fichier universel, utilisé par PHYSIO (et certains modules associés). Il contient les définitions du bloc de données des canaux (CDB), de la table de "dispatching" des canaux, du bloc de données des unités (UDB), de la table de "dispatching" des unités, des blocs de demande d'E/S (IORB).

PROLOG

C'est un fichier de parametre, d'assignements de mémoire, et de définitions de macros. Il définit les grandes régions de l'espace adresse du moniteur mais aussi les PSB et JSB.

STG

L'occupation de la mémoire du moniteur (résident et non résident) est définie ici.

A N N E X E D

QNAP : QUEUING NETWORK ANALYSIS PACKAGE

QNAP est un outil destiné à décrire, traiter et résoudre des modèles de files d'attente appelés souvent réseaux.

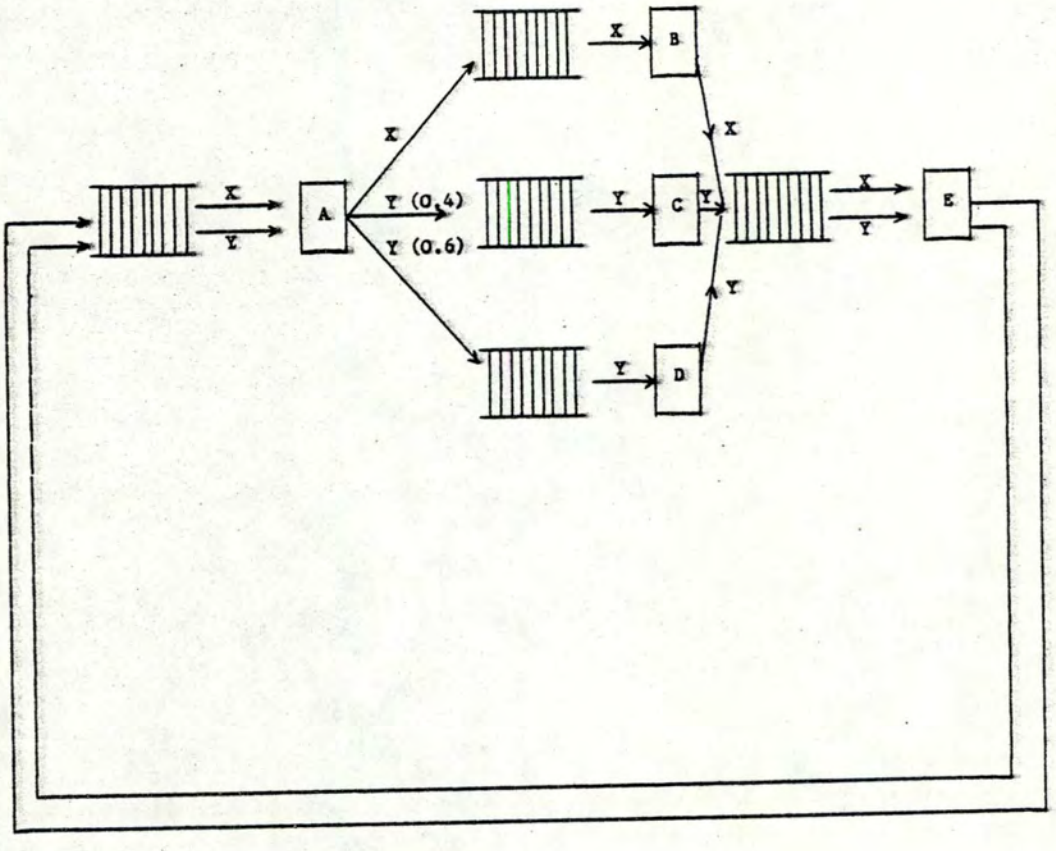
Notre but dans cette annexe est de décrire le stricte minimum pour être capable de comprendre et utiliser QNAP.

QNAP comporte un langage de spécification pour décrire le modèle et contrôler sa résolution, ainsi que plusieurs modules de résolution (SOLVERS) implementant différentes techniques.
Nous allons d'abord, dans la 1ère partie décrire les principaux mécanismes de modélisation, ensuite, la 2ème partie traitera du langage de spécification et nous terminerons dans la 3ème partie par les différents modules de résolution et leurs conditions d'application.

I. LES MECANISMES DE MODELISATION

Un réseau est constitué d'un ensemble de stations entre lesquelles circulent des clients. Ces clients peuvent appartenir à des classes différentes caractérisées par des comportements différents, des services différents dans les stations et des transitions différentes entre les stations.

La figure suivante, représente un tel réseau



Dans cette partie, nous allons expliquer les différents mécanismes permettant une telle modélisation en réseau.

Tout d'abord, les stations peuvent être de 4 types :

- serveur
- sémaphores (point I.9)
- ressources (point I.10)
- source (point I.11)

Ensuite, les stations de type serveur sont décrites par

- leur file d'attente (I.2)
- leur serveur (I.1, I.3, I.4)
- les niveaux de priorité des clients dans cette station (I.5)
- la transition vers une autre station en sortant de cette station (I.6)

Ajoutons qu'un client en cours de service peut en créer un autre (I.7)
Et n'oublions pas que l'on peut synchroniser les clients dans le système au moyen de flags (I.8).
Certaines possibilités ne sont significatives que pour des "SOLVERS" particuliers. Nous n'en tiendrons pas compte ici, nous donnerons plutôt une description globale des mécanismes.

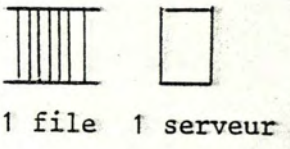
I.1. - Les serveurs

Un serveur effectue un service pour les clients de la station (service décrit plus loin). Quand le service est terminé, le client est envoyé vers une autre station selon les probabilités de transition. Le serveur sert alors le client suivant. S'il n'y en a pas, le serveur reste libre.
REMARQUE : Un client est servi par un seul serveur à la fois.

Il existe plusieurs types de serveurs, les voici :

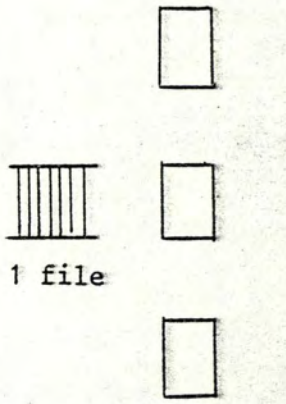
- serveur unique :

le serveur traite un seul client à la fois.



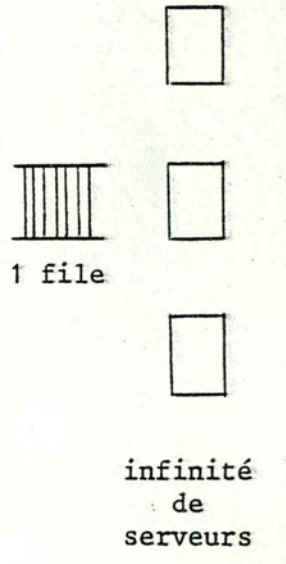
- serveur multiple (n) :

le serveur traite n clients à la fois. C'est en fait n serveurs absolument identiques associés à la même file.



- serveur infini :

le serveur peut servir tous les clients à la fois. Il agit comme si il était composé d'une infinité de serveurs identiques associés à la même file. Le client entrant dans une telle station trouve toujours un serveur libre et n'attend donc jamais.



I.2. - Organisation des files d'attente

La discipline de la file détermine quelle position un nouveau client doit prendre en arrivant dans la file. Les différentes disciplines possibles sont :

FIFO : le nouveau client est placé en fin de file

LIFO : il est placé en début de file

FIFO + PRIOR : chaque client a un niveau de priorité. Les clients sont placés dans la file selon leur priorité : la plus grande priorité en début de file.
Les clients de priorités égales sont placés selon la discipline FIFO

LIFO + PRIOR : idem mais les clients de priorités égales sont placés selon la discipline LIFO

En même temps que la discipline de la file, on doit prendre en compte la manière dont l'allocation du serveur est faite (point suivant).

I.3. - L'allocation du serveur

L'allocation du serveur est la manière dont on choisit dans la file, le prochain client à servir.
Les différents types d'allocation sont :

- Station sans quantum ni preemption

  Le serveur sera réalloué immédiatement après le départ du (ou des) client(s) courant(s). Quand un client entre dans la station, si il existe un serveur libre, il lui est immédiatement alloué sinon, il attend dans la file.
  Un client quitte la station quand son service est terminé ou quand un autre client l'oblige à partir. Le serveur est alors alloué au premier client de la file.

- Station avec préemption

Dès qu'un client de priorité supérieure à celle du client courant arrive dans la file, le service est interrompu, le client courant retourne dans la file et le serveur est immédiatement alloué au nouveau client.

La préemption peut être utilisée avec les disciplines LIFO et PRIOR.

- Station avec quantum

Le quantum est une quantité de temps. Quand le client a reçu du serveur une période de temps égale au quantum, il est envoyé à la fin de la file, et le client suivant est servi.

Le quantum (quantité) peut être différent par classe de clients.

Dans une même station, on ne peut pas utiliser une allocation par quantum en même temps qu'une allocation avec preemption et/ou une discipline PRIOR.

I.4. - Le service

Le service d'un client dans une station commence dès le moment où le serveur est alloué à ce client. Le service peut être différent par classe de clients. Ce service comprend deux choses :

- la demande de travail

C'est la quantité de travail demandée par un client dans une station. Cette quantité peut être donnée soit par une valeur constante soit par une distribution de probabilité.

La station peut travailler avec un taux de service spécifié (valeur par défaut = 1). Ce taux peut être considéré comme la vitesse à laquelle le serveur traite la demande de travail du client présent dans la station. Il peut être constant ou variable : dépendant du nombre de clients dans la file. Ce taux peut être différent par classe de clients.

La demande de travail du client divisée par le taux de service du serveur donne la période de temps passée par le client dans le serveur.

- Opérations de manipulation d'objets

Pendant son service, un client peut effectuer les opérations suivantes qui seront décrites plus loin. Il peut :

- forcer la transition d'un client vers une station (I.6)
- créer de nouveaux clients (I.7)
- tester ou établir l'état d'un flag (I.8)
- demander ou donner un laisser-passer sur un sémaphore (I.9)
- demander ou relacher une ressource (I.10)
- libérer un client en attente (I.8, I.9, I.10)

Ces opérations se font à l'intérieur d'un service, mais ne consomment aucun temps de service. Ce sont des opérations intantanées. Un client mis en état d'attente ne peut plus être servi : il est bloqué. Si un serveur est alloué à ce client bloqué, le serveur devient inactif mais toujours alloué à ce client bloqué.

Le service d'un client est terminé quand

- soit la demande de travail et les opérations de ce service sont terminées.
- soit quand le client demande pour quitter la station pendant son service ou qu'il est forcé par un autre client de la quitter. Dans ce cas, le service est interrompu et ne peut être repris.

I.5. - Le niveau de priorité des clients

Chaque client a un niveau de priorité qui peut varier durant sa vie dans le modèle.

On peut changer la priorité d'un client de différentes manières :

- le paramètre PRIOR d'une station

  Définit pour chaque classe, le niveau de priorité des clients entrant dans la station. Les classes de clients pour lesquelles aucun niveau de priorité n'est défini gardent leur priorité précédente.

- la procédure TRANSIT dans un service

  Force la transition du client effectuant ce service vers une station-destination et avec un niveau de priorité spécifié. La paramètre TRANSIT de la station que le client quitte et le paramètre PRIOR de la station où le client entre ne sont pas pris en considération.

- la procédure PRIOR dans un service

  Modifie la priorité d'un client de la file associée au service. Ce client peut être celui en cours de service ou un client en attente. Remarquons que si l'allocation du serveur se fait avec préemption, cela peut interrompre le service du client "courant" en faveur d'un client de priorité supérieure.

I.6. - Transition d'un client

La transition d'un client d'une station vers une autre station est déterminée de plusieurs manières.

- le paramètre TRANSIT d'une station

  Il définit la station-destination d'un client ayant fini son service dans cette station. Il peut y avoir plusieurs stations-destination différentes, mais, il faut alors spécifier la probabilité d'aller vers chacune d'entre elles.

- la procédure TRANSIT dans le service du client

  Un client effectuant un tel service force la transition d'un client d'une certaine file vers une autre file, avec la classe et la priorité spécifiée par la procédure TRANSIT. Si le client forcé à changer de file est en cours de service, le service est interrompu et ne peut être repris.

  Le paramètre TRANSIT de la station que le client quitte est ignoré.

- procédure MOVE dans le service d'un client

  Un client effectuant un tel service force la transition du 1er client d'une station désignée vers une autre station. Le paramètre TRANSIT de la station que le client quitte est ignoré.

I.7. - Création d'un client dans le service d'un autre client

Un client en cours de service peut en créer dynamiquement un autre qui aura les mêmes caractéristiques (priorité et classe) que son "père". Mais le fils n'est envoyé dans aucune file. Pour cela, il faut employer la procédure TRANSIT qui permet également de modifier la priorité du fils créé.

I.8. - Les flags

Le flag est un moyen de synchronisation des clients plus simple que les sémaphores et les ressources.

Il existe deux états possibles pour un flag :

set et reset

Les opérations qu'un client effectuant un service peut exécuter sur un flag sont :

- la création dynamique avec comme état initial du flag : reset
- mettre le flag dans l'état set : tous les clients attendant sur ce flag sont alors libérés.
- mettre le flag dans l'état reset
- forcer un client ou se forcer lui-même à attendre qu'un certain flag soit dans l'état set
- forcer un certain client (ou lui même) à quitter son état d'attente du flag. Ce client peut donc continuer son service, s'il n'est pas bloqué ailleurs.

I.9. - Les sémaphores

Un sémaphore à une fonction similaire au flag mais il a en plus un compteur.

Ce compteur est :

- soit le nombre de laisser-passers disponibles s'il est positif
- soit le nombre de clients en attente s'il est négatif.

Les différentes opérations possibles qu'un client effectuant un service peut faire sur un sémaphore sont :

- demander un laisser-passer
  s'il y en a encore, décrémenter le compteur d'une unité
  s'il n'y en a plus, le client est bloqué et placé dans la file associée au sémaphore
- rendre son laisser-passer et donc incrémenter le compteur
- détruire toutes les demandes d'un client sur un sémaphore, et donc le débloquer s'il n'est pas bloqué ailleurs.

I.10. - Les ressources

Une ressource est une station constituée d'une file et de une ou plusieurs unités de ressources (serveur passif).
Les différentes opérations qu'un client effectuant un service peut faire sur une ressource sont :

- demander une ressource :
  chaque client peut demander plusieurs ressources simultanément. Si la demande n'est pas satisfaite, le client est bloqué jusqu'à ce que la ressource soit disponible.
- relacher une ressource et donc la rendre disponible pour un autre client.
  Un client ne peut relacher une ressource que si il l'a demandée avant.
- détruire toutes les demandes d'un client sur une ressource.

Remarquons qu'un modèle constitué par un ensemble de serveurs peut être remplacé par un modèle symétrique constitué de ressources.

I.11. - Station source

Une station source est une station particulière qui génère des clients infiniment. L'intervalle de temps entre deux générations est défini par la demande de travail faite dans le service de la station (voir I.4).
Elle peut être considérée comme une station à serveur simple initialisée avec un nombre infini de clients dans sa file.

II. LANGAGE DE SPECIFICATION

Ce langage comporte 2 niveaux

- un langage de "contrôle" pour décrire le réseau
- un langage algorithmique qui ressemble au PASCAL et complète le langage de contrôle.

Nous nous limiterons ici à une description générale de ces deux langages ; visant plus ici une compréhension des possibilités de QNAP que de rendre capable d'utiliser syntaxiquement parlant QNAP. Une telle description syntaxique figure par ailleurs dans le "manuel de référence" de QNAP.
Toutefois, il faut remarquer que l'on peut utiliser le langage algorithmique sans pour autant utiliser le langage de "contrôle". Dans ce cas, QNAP est identique à un langage tel que PASCAL.

II.1. - Données manipulées par QNAP

Avant de passer à la description du langage en lui-même, il est bon de signaler ou de rappeler les différents types de données manipulés par QNAP.
Ces types de données sont répartis en 4 groupes

1) types scalaires

Ce sont des variables non-structurées qui sont

- des INTEGERs,
- des REALs,
- des BOOLEANs,
- ou des STRINGs.

L'utilisateur ne peut pas créer lui-même d'autres types scalaires.

2) les types tables ou variables dimensionnées

Ce sont des variables structurées.

Il s'agit de collections d'éléments du même type : soit scalaire
soit objet
soit référence

N.B. : les tables de tables ne sont pas permises.

Ces tables peuvent être à une ou plusieurs dimensions.

Type, dimension et longueur sont définis par l'utilisateur.

3) les types objets

Ce sont des variables structurées.

Il s'agit également de collections d'éléments mais contrairement aux tables les éléments d'un objet peuvent être de types différents :

- soit scalaires
- soit objets
- soit références
- soit tables.

Ces éléments sont appelés les attributs des objets.

- Il existe des types d'objets prédéfinis avec des attributs prédéfinis.

  Mais l'utilisateur peut leur ajouter des attributs qu'il déclarera lui-même.

  Ces types d'objets préféfinis sont QUEUE (la file de la station)
  CLASS (la classe du client)
  CUSTOMER (le client)
  FLAG

  exemples :

  QUEUE INTEGER N ; déclaration d'un attribut "INTEGER" pour les files

  CUSTOMER REAL A,B ; déclaration de 2 attributs "REAL" pour les clients

  QUEUE REAL STAT (5) ; déclaration d'un attribut étant une table de 5 "REALs".

- l'utilisateur peut également définir lui-même d'autres types d'objets en énumérant les attributs de ce type d'objet.

exemple :

```
OBJECT NODE (N,T) ;

        INTEGER N ;
        QUEUE CPU ;
        QUEUE DISK(N) ;
        REAL T ;
END ;
```

déclaration du type d'objet "NODE" ayant 2 paramètres formels (NT) et 4 attributs (N, CPU, DISK (N), T).

4) les types références

Ce sont des variables non structurées.

Une référence est caractérisée par le type d'objet qu'elle peut référencer.

exemple :

En début de programme on déclare

```
QUEUE CPU1, CPU2 ;
REF QUEUE Q ; la référence Q peut référencer le type d'objet QUEUE
```

Plus loin : dans le langage algorithmique

```
Q : = CPU1 ; Q référence la file CPU1
```

5) les listes

Les listes facilitent la désignation et la manipulation d'objets et de variables du même type.

exemples :

on a déclaré

```
INTEGER N = 10, L (N) ;
QUEUE DISK (20), CPU ;
REF QUEUE Q (30) ;
```

et dans le langage algorithmique

L : = 1 STEP 1 UNTIL N ; liste des n premiers entiers

L : = 1 REPEAT N ; liste de n "1"

Q : = ALL QUEUE ; liste de toutes les files déclarées.

II.2. - Le langage algorithmique

Ce langage est très proche du PASCAL et comporte

1) des instructions de déclaration
   - déclarations de variables et de tables
   - déclarations d'attributs
   - déclarations de types d'objets
   - déclarations d'objets
   - déclarations de références
   - déclarations de labels
   - déclarations de procédures
   - des initialisations statiques des données manipulées. Si certaines initialisations ne sont pas explicitées, il existe des valeurs par défaut :

     integers = 0

     reals = 0

     booleans : FALSE

     strings : empty

     références : NIL.

2) des expressions

   Ces expressions sont simples ou conditionnelles. Elles utilisent les opérateurs classiques :

   - opérateurs de relations :

     <, >, =, <>, <=, >=

- opérateurs mathématiques

  +, -, *, /, OR, AND, NOT

Ces expressions peuvent être

- des constantes,
- des variables,
- des appels de fonctions,
- ou des listes de données.

3) <u>des instructions</u>

Ces instructions sont à peu de choses près celles de PASCAL

- simples ou composées
- instructions d'affectation

- GO TO
- IF
- WHILE
- FOR
- appels à des procédures

1. certaines sont définies par QNAP

   * procédures d'impression des résultats.

     PRINT, PRINTF : pour écrire une ligne.

     OUTPUT : pour imprimer les résultats de la simulation du modèle.

   * sauvetage et restauration du modèle

     SAVE (S)

     RESTORE [(S)]

   * appels aux modules de résolution

     SOLVE, MARKOV, SIMUL

   * demande de travail (par le service d'une station)

     CST (a), EXP (a) ...

* procédures de synchronisation (voir concepts)

P ([c,] q [,k] [,i])

force le client "c" à demander la ressource ou le sémaphore associé à la file "q".

Cette demande est faite avec la priorité "i" et la classe "k".

Si "c" n'est pas spécifié, on prend le client courant, si "k" et/ou "i" ne sont pas spécifiés, on prend la classe et/ou la priorité du client.

V ([c] ,q)

relacher une ressource ou un sémaphore.

FREE (c, [q, f])

détruire les demandes sur une ressource ou un sémaphore.

WAIT ([c,] f)

attendre qu'un flag soit dans l'état set

SET (f)

RESET (f)

mettre un flag dans l'état set ou reset

PRIOR ([c,])

voir I.5

TRANSIT ([c,] q [,k] [,i])

voir I.5, I.6

MOVE (q1, q2)

voir I.6

2. l'utilisateur peut aussi définir lui-même des procédures.

- appel à des fonctions prédéfinies

* fonctions mathématiques

ABS (a), FIX (a) ...

* génération "random"

EXP (a), HEXP (a) ...

* accès aux résultats à la fin de la résolution

MSERVICE, MBUSYPCT ...

* création d'objets

NEW (a)

II.3. - Le langage de contrôle

Un programme QNAP comprend différentes parties fonctionnelles introduites par les différentes commandes.

/DECLARE/ : déclaration des identifiants
(déclarations du langage algorithmique)
/STATION/ : description des caractéristiques d'une station
/CONTROL/ : spécification des paramètres de contrôle
/EXEC/ : algorithme d'exécution : instructions du langage algorithmique
/TERMINAL/ : exécution interactive du programme : les commandes suivantes sont lues à partir du terminal
/RESTART/ : introduction d'un nouveau modèle et remise à l'état initial de QNAP
/END/ : fin du fichier source QNAP.

Les commandes DECLARE et EXEC reprennent les instructions du langage algorithmique.
Les commandes TERMINAL, RESTART et END ne comportent ni instructions, ni parametres. Il ne reste plus qu'à détailler les commandes STATION et CONTROL.

N.B. : En utilisant uniquement les commandes DECLARE et EXEC, on a à notre disposition un langage traditionnel (tel PASCAL), tandis qu'en utilisant toutes les commandes, on définit un réseau.

exemple 1 :

```
/DECLARE/    QUEUE A,B ;
/STATION/    NAME = A ;
             INIT = 1 ;          nombre de clients dans la station A

/CONTROL/    TMAX = 1000. ;      durée de la simulation
/EXEC/       SIMUL ;             1ère simulation avec un client pen-
                                 dant une durée de 1.000

/STATION/    NAME = A
             INIT = 3 ;
/EXEC/       SIMUL ;             2ème simulation avec trois clients
                                 pendant une durée de 1.000

/CONTROL/    TMAX = 5000. ;
/EXEC/       SIMUL ;             3ème simulation avec trois clients
                                 pour une durée de 5.000
```

exemple 2 :

fichier source

```
/STATION/    NAME = A ;          description du modèle
     :
     :
/TERMINAL/ ;                     le reste est lu à partir du terminal
```

entré au terminal

```
/EXEC/       SOLVE ;             demande d'une solution analytique

/STATION/    NAME = A ;
             SERVICE = EXP (5) ;

/EXEC/       SOLVE
```

exemple 3 :

```
/DECLARE/     QUEUE A, B, C ;
              CLASS X, Y;
/STATION/     NAME = A ;
                 :
/EXEC/        SIMUL ;              résolution d'un modèle

/RESTART/
/DECLARE/     QUEUE A, B, C, D ;

/EXEC/        SOLVE ;              résolution d'un deuxième modèle
                                   n'ayant plus rien à avoir avec
                                   le premier

/END/
```

1) /STATION/

Le but de cette commande est de décrire une ou plusieurs stations du modèle.

Voici les différents paramètres de cette commande

- NAME = identifiant de la file ou liste d'identifiants de files.

exemple :

```
/DECLARE/     QUEUE  CPU, DK(10)
/STATION/     NAME = CPU ;
                 :
/STATION/     NAME = DK (1 STEP 1 UNTIL 5)
                 :                 définition de stations associées
                 :                 aux 5 premières files de DK (10)
                 :
/STATION/     NAME = CPU            altération de la définition précé-
                 :                 dente de la station CPU
                 :
```

- TYPE = spécifie le type de la station.

TYPE = * SERVEUR [,multiple]
* RESSOURCE [,multiple]
* SEMAPHORE [,multiple]
* SOURCE

où multiple = * SINGLE
* MULTIPLE (integer)
* INFINITE

- SCHED = précise la discipline et l'algorithme de gestion de la file.

SCHED = order [,PRIOR] [,PREEMPT]
/PRIOR [,PREEMPT]
/QUANTUM [(real)]

order = FIFO/LIFO FILO/PS

PS : "Processor sharing" : tous les clients sont servis ensemble avec un taux de service divisé par n

n = nombre de clients dans la station.

exemple :

```
/DECLARE/   QUEUE UC, DK, CHANNEL ;
            INTEGER   NPROC ;

/STATION/   NAME = UC ;
            TYPE = MULTIPLE (NPROC) ; (NPROC = nombre de serveurs de
            SCHED = LIFO, PREEMPT ;            la station uc)
              :
              :
/STATION/   NAME = DK ;                (par défaut, ce serveur est un
              :                         serveur "SIMPLE" et a une
              :                         discipline de file "FIFO")
              :
/STATION/   NAME = CHANNEL ;
            TYPE = RESSOURCE ;
            SCHED = QUANTUM (0,2)
```

- INIT (class) = integer

  initialise le nombre de clients dans la file, par classe.

exemple :

```
/DECLARE/     QUEUE A, B ;
              CLASS X, Y ;
              INTEGER N ;

/STATION/     NAME = A ;
              INIT (X) = 2 ;   INIT (Y) = 4 ; (nombre de clients et
                                              dans les classes X et Y)

/STATION/     NAME = B
              INIT = N
```

- SERVICE (class) = statement

  Spécifie par classe de clients le service que le serveur doit effectuer.

  Ce service est composé de deux choses :

  - une demande de travail selon une distribution de probabilité
  - des manipulations d'objets et des procédures spéciales.

exemple :

```
/DECLARE/     QUEUE A, SEM ;
              CLASS X, Y, Z ;
              REAL SA ;

/STATION/     NAME = A ;
              SERVICE (X) = CST (4)
              SERVICE (Y) = HEXP (SA, 16)
              SERVICE = BEGIN                  (définit le service des
                          EXP (12.) ;           clients de classe Z)
                          P (SEM) ;
                          PRINT (TIME) ;
                          CST  (5.) ;
                          V (SEM) ;
                        END ;
```

- TRANSIT (class) = transition - list

ATTENTION : à ne pas confondre avec la procédure TRANSIT que l'on peut effectuer à l'intérieur d'un service.

Ce paramètre décrit pour les clients sortant de cette station et par classe de clients, les stations destinations et les probabilités de transition associées.

exemple :

```
/STATION/     NAME = A ;
              SERVICE =
                   !
                   !
              TRANSIT (X) = A, 0.5, B, 0.3, C ;
```

en utilisant des probabilités relatives

```
              TRANSIT (X) A, 10, B, 6, C, 4 ;
```

A la fin de son service, le client de classe X va vers la station

- A avec une probabilité 0.5
- B avec une probabilité 0.3
- C avec une probabilité 0.2

```
              TRANSIT (Y) = A, Z, 1, C, Y, 2, OUT, 1 ;
```

A la fin de son service, le client de classe Y va vers la station

- A avec une prob 1/4 et une priorité Z
- C avec une prob 2/4 et une priorité Y
- OUT avec une prob 1/4

- RATE (class) = real

spécifie la vitesse à laquelle le ou les serveurs de la station travaillent.

Il s'agit du nombre d'unités de travail accomplies pendant une unité de temps. Ce taux peut être différent pour chaque classe.

exemple :

```
/STATION/     NAME = A ;
              SERVICE = EXP (5) ;
              RATE = 2.0 ;
              RATE (X) = RX ;
```

- PRIOR (class) = integer

  ATTENTION : à ne pas confondre avec la procédure PRIOR que l'on peut effectuer à l'intérieur d'un service.

  Ce paramètre détermine le niveau de priorité des clients entrant dans la station. Il peut être différent selon les classes.

exemple :

```
/STATION/      NAME = UC ;
               PRIOR (X, Y) = PX ;
               PRIOR = 2 ;

/STATION/      NAME = UK ;
               SCHED = PRIOR, PREEMPT ;
               PRIOR (Y) = 1 ;
               PRIOR (Z) = 3 ;
               :
               :
               :
```

- QUANTUM (class) = real

  Alloue par classe une tranche de temps dans la station.

  Le QUANTUM du paramètre SCHED est considéré comme une valeur de défaut si le paramètre QUANTUM n'est pas spécifié.

exemple :

```
/STATION/      NAME = UC ;
               SCHED = QUANTUM (0.3) ;
               QUANTUM (X) = QT ;
```

est équivalent à

```
/STATION/      NAME = UC ;
               SCHED = QUANTUM ;
               QUANTUM (X) = QT ;
               QUANTUM = 0.3 ;
```

- COPY = Queue

Ce paramètre sert à copier les paramètres d'une autre station.

exemple :

```
/STATION/     NAME = DKO
                  :
                  :
/STATION:     NAME = DK1 : COPY DKO
/STATION/     NAME = DK2 : COPY DKO ; PRIOR (Y) = O ;
```

PRIOR remplace le paramètre correspondant de DKO, ou s'y ajoute.

N.B. : (class) est chaque fois facultatif et peut être une liste de classes.

2) /CONTROL/

Cette commande sert à modifier les valeurs par défaut des paramètres internes de QNAP.

Voici les différents paramètres

- paramètres qui contrôlent les E/S

OPTION : paramètre qui contrôle l'impression des résultats.

Les différentes options sont SOURCE/NSOURCE
RESULT/NRESULT
TRACE/NTRACE
DEBUG/NDEBUG
VERIF/NVERIF

UNIT : paramètre qui contrôle les différents numéros d'unité logique associés aux fichiers d'entrée/sortie de QNAP.

- paramètres qui contrôlent la réduction

CLASS = queue liste

demande des statistiques sur les classes des stations spécifiées.

MARGINAL = queue-list

demande les probabilités marginales (globalement ou pour chaque classe) pour les stations spécifiées.
Prob marginale pour une station = Prob que cette station contienne 0, 1 ... n clients.

ENTRY = statements

décrit une séquence d'instructions qui doit être exécutée avant chaque résolution du modèle.
ex : - initialisation de variables utilisées dans /STATION/
- impressions .

EXIT = statements

décrit une séquence d'instructions qui doit être exécutée après chaque résolution du modèle.
ex : - calcul de résultats supplémentaires
- impressions.

CONVERGENCE = real-list

modifie la valeur par défaut de certains paramètres de SOLVE et MARKOV.

- paramètres qui contrôlent la simulation

TMAX = real

contrôle la durée maximum de la simulation.

PERIOD = real

donne la période d'activation de la séquence test (voir paramètre TEST)

RANDOM = integer

donne une valeur de départ au générateur congruentiel.

TRACE = real [,real]

donne le moment où l'on va commencer et le moment où l'on va terminer à garder la trace de tous les événements, garder tous les résultats intérmédiaires.

ACCURACY = queue list

spécifie les stations pour lesquelles il faut calculer l'intervalle de confiance.

TEST = statements

décrit une séquence d'instructions à exécuter à intervalles réguliers pendant la simulation.

CORRELATION = queue list

demande, pour les files spécifiées la vérification des mesures faites pendant la simulation.

- paramètres divers

NMAX = integer

utilisé pour augmenter le nombre maximum de classes disponibles dans un modèle.

ALIAS : {(ident, ident)} [.....]

donne la possibilité de donner un identifiant alias à un identifiant existant déjà.

- exemple :

```
/DECLARE/   REAL
/CONTROL/   OPTION = NSOURCE, NRESULT
            UNIT = OUTPUT (6)
            CLASS = NIL
            MARGINAL = ALL QUEUE
            TMAX = T
/DECLARE/   :
/STATION/   :
/EXEC/      :
```

III. LES MODULES DE RESOLUTION

Nous allons traiter ici des différents modules de résolution de QNAP et de leurs conditions d'application.

Les modules de résolution sont appelés dans la commande EXEC sous la forme
- SOLVE ("nom de solveur")
- MARKOV
- SIMUL

A la fin de la résolution, les résultats sont imprimés dans un rapport standard, selon les paramètres OPTION, CLASS, et MARGINAL de la commande /CONTROL/ (voir II.3).

Les différents résultats imprimés dans le rapport standard sont pour chaque station
- le facteur d'utilisation
  c-a-d le pourcentage de temps pendant lequel le serveur (ressource) est occupé.
  Il est égal à 0 dans le cas d'une station infinie.
- le temps moyen de service
  c-a-d le temps moyen de service tel que le voit le client.
- le temps moyen de réponse
  c-a-d la somme du temps d'attente et du temps de service.
- le nombre moyen de clients dans la station
- le "THROUGHPUT" moyen
  c-a-d le nombre moyen de clients servis par unité de temps.

Rappelons que l'on peut obtenir ces informations, ainsi que d'autres au moyen de fonctions (voir langage algorithmique).

A ce rapport s'ajoutent les probabilités marginales et la trace, si on les a demandées.

III. 1. - Les "solvers" analytiques SOLVE

Les solvers analytiques sont des modules de résolution approximatifs ou exacts. Leurs temps de résolution sont toujours inférieurs à ceux du MARKOV ou SIMUL.

- conditions d'application

Les solvers analytiques ne sont pas applicables à tous les types de réseaux.

Certaines restrictions doivent être faites sur plusieurs points.

1) les transitions :

Elles peuvent être différentes par classes.

La classe d'un client peut être modifiée pendant une transition.

Les probabilités de transition ne peuvent pas dépendre de l'état du système.

2) les services :

Ils sont réduits à une simple demande de travail. Les procédures de synchronisation sont interdites. Ils ne peuvent pas dépendre de l'état du réseau.

3) les types de stations :

Il ne peut pas y avoir de stations ressources, de sémaphores, de flags et de files sans service.

4) le scheduling :

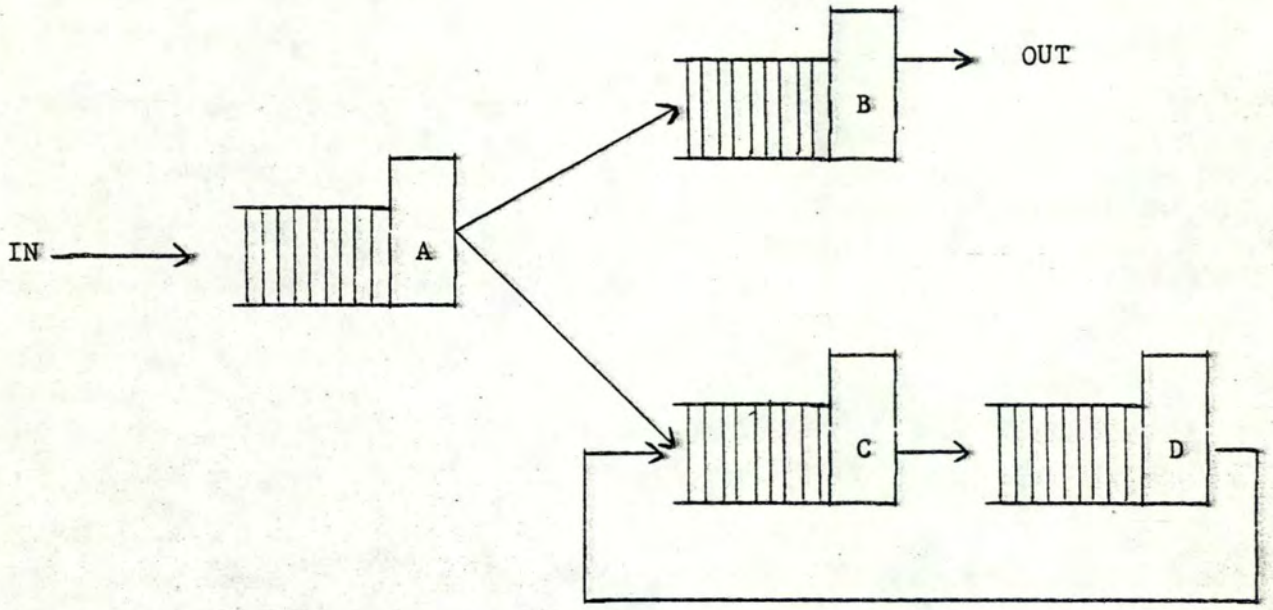
Le quantum n'est pas permis.

5) les réseaux :

* Définitions :

1 sous-chaine = un ensemble de paires (station, classe) qui communiquent entre elles selon les transitions définies dans le modèle.

exemple :



A - B est une sous-chaine

A - C - D - C est une sous-chaine

C - D - C est une sous-chaine.

1 sous-chaine fermée = 1 sous-chaine dont la population est constante. Une telle sous-chaine ne contient pas de station-source ni de file OUT.

exemple : C - D - C

1 sous-chaine ouverte = 1 sous-chaine qui contient au moins une station source et une file OUT.

exemple : A - B ; A - C - D - C

1 réseau fermé = 1 réseau dont toutes les sous-chaines sont fermées.

1 réseau ouvert = 1 réseau dont toutes les sous-chaines sont ouvertes

1 réseau mixte = 1 réseau qui contient au moins une sous-chaine ouverte et au moins une sous-chaine fermée.

* Restrictions :

1 sous-chaine ouverte doit contenir seulement une station-source.

1 réseau ouvert doit avoir soit une seule station-source pour toutes les sous-chaines

soit une station source différente pour chaque sous-chaine.

1 réseau mixte doit avoir une station source différente pour chaque sous-chaine ouverte.

1 station source ne peut pas envoyer les clients directement dans une file OUT.

1 classe de clients donnée doit appartenir à une seule sous-chaine.

ergodicité : pour chaque paire (station A, classe X), le client quittant la station A avec la classe X, doit y retourner avec la même classe après une période finie.

- les différents "solvers" analytiques

Voici la liste des différents solvers analytiques dont QNAP dispose. Certains solvers peuvent avoir des conditions d'application plus restreintes que celles énumérées ci-dessus.

· CONVOL (algorithme de convolution)

MVA (mean value analysis)

MVANCA (mean value analysis and normalized, convolution algorithms)

HEURSNC (schweitzer, Neuse and chandy heuristic algorithms)

PRIROPR (Veran algorithm for preemptive priority scheduling)

ITERATIV (Marie iterative algorithms)

DIFFU (Gelembe and Pujalle diffusion algorithms).

III. 2. - Le solver makovien MARKOV

Ce module donne une résolution théorique des réseaux. Il est exact. Mais consomme énormément de temps CPU.

- conditions d'application

1) le réseau doit être fermé, il peut y avoir une ou plusieurs classes de clients.
2) les stations permises sont :
   serveurs simples, multiples, infinis, files simples (sans services)
   Tandis que les ressources et sémaphores ne sont pas permis.
3) les schedulings permis sont :
   FIFO, LIFO avec ou sans priorités
   avec ou sans preemption
   PS
   Ici aussi, le quantum n'est pas permis.
4) les services :
   Une seule demande de travail est permise. Elle peut être précédée ou suivie d'instructions définissant certaines dépendances. Le taux de service peut dépendre du nombre de clients dans chaque station (globalement ou pour chaque classe)
   ex : SERVICE : IF CUSTNB (A) < 3
   THEN EXP (2)
   ELSE ERLANG (2,3)
5) les transitions :
   les procédures TRANSIT et MOVE peuvent être utilisées. Elles doivent apparaître dans la description du service après une procédure de demande de service.
6) le taux de service de chaque serveur d'une station peut dépendre du nombre de clients dans cette station.
7) les transitions instantanées :
   Après avoir effectué une transition à la fin d'un service il est possible d'initialiser d'autres transitions instantanées, sous des conditions dépendant du nombre de clients dans chaque station.

III. 3. - La simulation SIMUL

C'est une simulation en événement discret du modèle étudié. Cette façon de résoudre un modèle prend beaucoup de temps CPU. Mais plus la simulation est longue, meilleurs seront les résultats.

Ici, tous les mécanismes peuvent être utilisés excepté :

- le scheduling PS
- le taux de service dépendant du nombre de clients dans la file.

A N N E X E E

LE PROGRAMME DU MESUREUR

```
        TITLE PERFO

        search monsym,macdef,symdef


RELTEMP


;*** DEFINITION DES ACs ***

t1=1
t4=4
a1=5
a2=6
fx=7
p3=12
p4=13
p=17


        reloc 3000

;************************************************
;*** ROUTINES ET DONNEES DE     S N O O P      ***
;************************************************

usewr==1        ; Page utilisateur ou les routines de mesure sont ecrites.
userel: 1       ; Page ou les routines de mesure sont relosees

monpag: block 1 ;Page moniteur ou les routines sont 'lockees'

lock:                       ;* INSERTION DES PAGES DANS LE MONITEUR *

        JSY snoop,%I(.snplc),1,%I(usewr),-,<ercal errtrt>
        movem 2,monpag
        RET

defbkpt:                    ;* DEFINITION DES BREAK POINTS *

        SNPATC (atterm,newst,2,,<ercal errtrt>)
        SNPATC (entcpu,skdi2,,,<ercal errtrt>)
        SNPATC (cputer,wtcon2,2,,<ercal errtrt>)
        SNPATC (cputer,wtcon2,7,,<ercal errtrt>)
        SNPATC (cpucpu,schupc,3,,<ercal errtrt>)
        SNPATC (cpusol,ajbs0,1,,<ercal errtrt>)
        SNPATC (cpusol,ajbl69,1,,<ercal errtrt>)
        SNPATC (cpuuni,sio1,11,,<ercal errtrt>)          ; dec 9 ,sio1+11=sio2
        SNPATC (debpos,strtp1,6,,<ercal errtrt>)
        SNPATC (finpos,int2,,,<ercal errtrt>)
        SNPATC (finpos,phypdn,1,,<ercal errtrt>)
        SNPATC (posnul,sio4,1,,<ercal errtrt>)
        SNPATC (posnul,sek1,10,,<ercal errtrt>)          ;dec 8
        SNPATC (posnul,pos1,6,,<ercal errtrt>)
        SNPATC (posnul,pdn1,5,,<ercal errtrt>)
        SNPATC (debtra,strti1,26,,<ercal errtrt>)        ; dec 22
        SNPATC (fintra,intdon,3,,<ercal errtrt>)
        SNPATC (tranul,strti0,2,,<ercal errtrt>)
        SNPATC (tranul,xfrchb,14,,<ercal errtrt>)        ;dec 12
        SNPATC (tranul,unick8,62,,<ercal errtrt>)        ;dec 50
        SNPATC (trponu,strtpf,11,,<ercal errtrt>)        ;dec 9
        SNPATC (abtrek,rst1,4,,<ercal errtrt>)
        SNPATC (debhbl,xfrchb,10,,<ercal errtrt>)        ;dec 8
        SNPATC (debhbl,unick8,56,,<ercal errtrt>)        ;dec 46
        SNPATC (tstipo,uniov0,3,,<ercal errtrt>)
        SNPATC (tstitr,uniov0,5,,<ercal errtrt>)
        SNPATC (eliorb,donirb,20,,<ercal errtrt>)        ;dec 16

        RET
```

```
addmon:                         ;* ACQUISITION DES ADRESSES DU MONITEUR *

        SNPADD (todclk,,,<ercal errtrt>)
        SNPADD (fkstat,,,<ercal errtrt>)
        SNPADD (tcotst,,,<ercal errtrt>)
        SNPADD (ttobet,,,<ercal errtrt>)
        SNPADD (tcitst,,,<ercal errtrt>)
        SNPADD (fkq2,,,<ercal errtrt>)
        SNPADD (cst1,,,<ercal errtrt>)
        SNPADD (cst5,,,<ercal errtrt>)
        SNPADD (strtab,,,<ercal errtrt>)

        RET

mreloc:                         ;* RELOCATION DES ROUTINES DE MESURE *

        move 7,monpas           ;calcul de la difference d'adressage
        sub 7,userel
        imuli 7,1000

        FOR 6,%I(debmon),%I(endmon),1

                hrrz 1,(6)

                IF <GTI 1,p>            ;ne pas reloger les ac
                THEN addm 7,(6)
                FI

        OD

        FOR 6,%I(debabs),%I(endabs),1   ; ne pas reloger les instructions a
                                        ; adresses absolues
                hrrz 4,(6)
                hrrz 1,(4)

                IF <GTI 1,p>
                THEN move 2,7
                     exch 2,(4)
                     subm 2,(4)
                FI
        OD

        move 1,monpas
        movem 1,userel

        RET

minser:                         ;* INSERTION DES BREAK POINTS *

        JSY snoop,%I(.snpih),-,-,-,<ercal errtrt>
        RET

mrembp:                         ;* ENLEVEMENT DES BREAK POINTS *

        JSY snoop,%I(.snprb),-,-,-,<ercal errtrt>
        RET

munloc:                         ;* LIBERATION DE L'ESPACE DANS LE MONITEUR *

        JSY snoop,%I(.snpul),-,-,-,<ercal errtrt>
        RET

;*** LISTE DES ADRESSES DES INSTRUCTIONS A ADRESSE ABSOLUE ***

debabs=.

endabs=.-1
```

```
;*****************************************
;*** PAGE A INSERER DANS LE  MONITEUR ***
;*****************************************

        loc 1000*useur

;*** DECLARATION DES DONNEES ***


saveac: block 3          ; zone de sauvetage des accumulateurs

moyter: block 6          ; moyenne de l'attente aux terminaux
cptter: block 6          ; compteurs du nbre d'attentes aux terminaux

fkstat: 0                ; adresse du moniteur
tcotst: 0                ; adresse du moniteur
ttobet: 0                ; adresse du moniteur
tcitst: 0                ; adresse du moniteur
fka2: 0                  ; adresse du moniteur
todclk: 0                ; adresse du moniteur
cst1: 0                  ; adresse du moniteur
cst5: 0                  ; adresse du moniteur
strtab:0                 ; adresse du moniteur

moycpu: block 6          ; moyenne du temps passe dans le CPU
varcpu: block 6          ; variance du temps passe dans le cpu
debcpu: 0                ; moment de l'entree dans le CPU
cptcpi: 0                ; nombre d'entrees dans le CPU
numfil: 0                ; numero de la file auxiliaire
cptcpo: block 6          ; nombre de sorties du cpu (par file)
outter: block 6          ; prob de trans terminaux (par file)
outcpu: block 6          ; prob de trans CPU (par file)
outsol: block 6          ; prob de trans GO-LISTE (par file)
outall: block 6          ; prob de trans unites (par file)
outuni: block 6          ; pourcentage vers chaque unite

ps: 0                    ; strins
us: 0                    ; strins

debupo: block 6          ; moment du debut du positionnement
cptpos: block 6          ; nombre de positionnements
moypos: block 6          ; moyenne du temps de pos
varpos: block 6          ; variance du temps de pos
cpscan: block 6          ; nombre de chansts PWQ-->TWQ
debutr: block 6          ; moment du debut du transfert
cpttra: block 6          ; nombre de transferts
moytra: block 6          ; moyenne du temps de trans
vartra: block 6          ; variance du temps de trans
abortr: block 6          ; nbre de suppression des demandes d'une unite
cpthbl: block 6          ; nbre de verifications du HB
cptirb: block 6          ; nbre d'IORBs ajoutes lors de l'elimination d'un IORB

ptuni: 0                 ; byte pointer du numero de l'unite
ptfil: 0                 ; byte pointer du numero de file



debmon:                  ; debut des routines a inserer dans moniteur

;*** APPEL AUX ROUTINES DE MESURE ***


atterm: CALL atermO
cputer: CALL cpterO
entcpu: CALL encpuO
cpucpu: CALL cpcpuO
cpusol: CALL cpsolO
cpuuni: CALL cpuniO
debpos: CALL debpoO
finpos: CALL finpoO
posnul: CALL posnuO
debtra: CALL debtrO
fintra: CALL fintrO
tranul: CALL tranuO
trponu: CALL trponO
abtrek: CALL abreaO
debhbl: CALL debhbO
tstipo: CALL tstipO
tstitr: CALL tstitO
eliorb: CALL elirbO
```

```
; *** MESURE DE L'ATTENTE AU TERMINAL , INSERER A NEWST + 2 (SCHED) ***

atermO: dmovem 2,saveac

        move 2,fkstat           ;)
        add 2,fx                ;)
        hrrz 2,0(2)             ;)verifie si l'attente etait bien due a une
        camn 2,tcotst           ;) E/S sur terminal
        jumpa aterm1            ;)oui
        camn 2,ttobet           ;)
        jumpa aterm1            ;)oui
        camn 2,tcitst           ;)
        jumpa aterm1            ;)oui
                                ;)non
        dmove 2,saveac
        RET
                                ; t1 contient le tps d'attente au terminal
aterm1: move 2,fkq2
        add 2,fx
        ldb 3,ptfil             ; determination du no de la file auxiliaire

        addm t1,moyter(3)       ;) calcul moyenne (par file)
        aos cptter(3)           ; compteur du nbre de mesures (par file)

        dmove 2,saveac
        RET
```

```
;*** MESURE DE L'ENTREE DANS LE CPU , INSERER A SKDJ2 (SCHED) ***

encpu0: movem 2,saveac

        move 2,@todclk          ; saisie du temps en millisecondes
        movem 2,debcpu

        aos cptcpi              ; compteur du nbr de proces entrant dans le cpu

        move 2,fka2
        add 2,fx
        ldb 2,ptfil             ; determination du no de la file auxiliaire

        movem 2,numfil          ; et memorisation

        move 2,saveac
        RET


;*** SORTIE CPU--->TERMINAUX & PROB DE TRANS , INSERER A WTCON2+2 ET A WTCON2+7
;                                                                  (SCHED) ***

cpter0: dmovem 2,saveac

        skipn debcpu
        jrst cpter9

        move 2,@todclk          ; saisie du temps en millisecondes

        move 3,numfil           ; saisie du no de file auxiliaire

        sub 2,debcpu            ;)
        addm 2,moycpu(3)        ;) calcul de la moyenne et de la variance
        imul 2,2                ;)   du temps passe dans le CPU (par file)
        addm 2,varcpu(3)        ;)
        aos cptcpo(3)           ; cpt du nbre de proc sortant du CPU (par file)

        aos outter(3)           ; probabilite de transition (par file)

cpter9: dmove 2,saveac
        RET


;*** SORTIE CPU---->CPU & PROB DE TRANSITION , INSERER A SCHUPC+3 (SCHED) ***

cpcpu0: dmovem 2,saveac

        skipn debcpu
        jrst cpcpu9

        move 2,@todclk          ; saisie du temps en millisecondes

        move 3,numfil           ; saisie du no de file auxiliaire

        sub 2,debcpu            ;)
        addm 2,moycpu(3)        ;) calcul de la moyenne et de la variance
        imul 2,2                ;)   du temps passe dans le CPU (par file)
        addm 2,varcpu(3)        ;)
        aos cptcpo(3)           ; cpt du nbre de proc sortant du CPU (par file)

        aos outcpu(3)           ; probabilite de transition (par file)

cpcpu9: dmove 2,saveac
        RET


;*** SORTIE CPU-----><OLISTE & PROB DE TRANS INSERER  A AJBSO +1 ET AJBL69+1 ***
;                                                          (SCHED)

cpsol0: dmovem 2,saveac

        skipn debcpu
        jrst cpsol9

        move 2,@todclk          ; saisie du temps en millisecondes

        move 3,numfil           ; saisie du no de file auxiliaire

        sub 2,debcpu            ;)
        addm 2,moycpu(3)        ;) calcul de la moyenne et de la variance
        imul 2,2                ;)   du temps passe dans le CPU (par file)
        addm 2,varcpu(3)        ;)
        aos cptcpo(3)           ; cpt du nbre de proc sortant du CPU (par file)

        aos outsol(3)           ; probabilite de transition (par file)

cpsol9: dmove 2,saveac
        RET
```

```
;*** SORTIE CPU--->UNITES & PROB DE TRANS , INSERER A SIO2 (PHYSIO) ***

cpuni0: dmovem 2,saveac
        movem 4,saveac+2

        skipn debcpu
        jrst cpuni9

        move 3,numfil           ; saisie du no de file auxiliaire

        move 2,@todclk          ; saisie du temps en millisecondes
        sub 2,debcpu            ;)
        addm 2,moycpu(3)        ;) calcul de la moyenne et de la variance
        imul 2,2                ;)    du temps passe dans le cpu (par file)
        addm 2,varcpu(3)        ;)

        aos cptcpo(3)           ; cpt du nbre de proc sortant du cpu (par file)
        aos nutall(3)           ; probabilite de transition (par file)

cpuni9: hll 2,0(p3)             ; (p3)= udbsts
        txne 2,1B16             ; disque ou bande
        jumpa cpmta1            ; cas d'une bande
                        ; determination de l'unite disque

        hrrz 2,12(p3)           ; 12(p3) = udbstr ------> ac2= nr de la structure
        add 2,strtab
        move 2,(2)              ; ac2= adr du sdb
        hlrz 3,12(p3)           ;    ------> ac3= nr de l'unite ds la structure

cpdsk1: move 2,0(2)             ; 0(2) = sdbnam
        camn 2,ps
        jumpa cpdsk4            ; structure = ps
        camn 2,us
        jumpa cpdsk5            ; structure = us
cpdsk2: movei 3,4               ; ni ps ni us

cpdsk3: dpb 3,ptuni             ; memorisation de l'unite ds 1 partie de UDBSTS
        aos outuni(3)           ; pourcentage pour chaque unite

        move 4,saveac+2
        dmove 2,saveac
        RET


cpdsk4: caig 3,2                ; structure = us
        jumpa cpdsk3
        jumpa cpdsk2


cpdsk5: caie 3,0                ; structure = us
        jumpa cpdsk2
        movei 3,3
        jumpa cpdsk3


                        ; determination de l'unite bande

cpmta1: movei 2,5
        dpb 2,ptuni             ; memorisation de l'unite ds 1 partie de UDBSTS
        aos outuni(2)           ; pourcentage pour chaque unite

        move 4,saveac+2
        dmove 2,saveac
        RET
```

```
;*** MESURE DU DEBUT DU POSITIONNEMENT, INSERER A STRTP1+6 (PHYSIO) ***

debpo0: dmovem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        move 3,@todclk
        movem 3,debupo(2)

        dmove 2,saveac
        RET


;*** MESURE DE LA FIN DU POSITIONNEMENT, INSERER A INT2  et PHYPDN (PHYSIO) ***

finpo0: dmovem 2,saveac


        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        skipn debupo(2)
        jrst finpo9

        move 3,@todclk
        sub 3,debupo(2)
        addm 3,moypos(2)
        imul 3,3
        addm 3,varpos(2)
        aos cptpos(2)

finpo9: dmove 2,saveac
        RET


;*** MESURE DU NOMBRE DE CHANGEMENTS PWQ-->TWQ INSERER A SIO4+1,A SEK1+8  ***
;                                          A POS1+6 ET A PDN1+5 (PHYSIO)  ***

posnu0: movem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        skipn cptpos(2)
        jrst posnu9

        aos cpscan(2)

posnu9: move 2,saveac
        RET


;*** MESURE DU DEBUT DU TRANSFERT,INSERER A STRTI1+22 (PHYSIO) ***

debtr0: dmovem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        move 3,@todclk
        movem 3,debutr(2)

        dmove 2,saveac
        RET


;*** MESURE DE LA FIN DU TRANSFERT,INSERER A INTDON +3,(PHYSIO) ***


fintr0: dmovem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

fintr1: skipn debutr(2)
        jrst fintr9

        move 3,@todclk
        sub 3,debutr(2)
        addm 3,moytra(2)
        imul 3,3
        addm 3,vartra(2)
        aos cpttra(2)

fintr9: dmove 2,saveac
        RET
```

```
;*** MESURE DU TRANSFERT NUL (CAS D'ERREUR )  ***
;       INSERER A STRTIO +2 (PHYSIO) A XFRCHB +12 ET A UNICK8 +50  ***

trannu0: movem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        aos cpttra(2)

        move 2,saveac
        RET


;*** MESURE DU TRANSFERT ET POSITIONNEMENT NULS (CAS D'ERREUER)
;                                       INSERER A STRTPF+9 (PHYSIO)   ***

trpon0: movem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        aos cptpos(2)
        aos cpttra(2)

        move 2,saveac
        RET


;*** MESURE DU NBRE DE SUPPRESSIONS DE TOUTES LES DEMANDES POUR UNE UNITE ***
;               INSERER A RST1 +4 (PHYSIO)                                  ***


abreq0: movem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        aos abortr(2)

        move 2,saveac
        RET


;*** MESURE DU NOMBRE DE VERIFICATIONS DU 'HOME BLOCK',
;               (AJout d'un IORB en tete de twq)
;                       INSERER A XFRCHB +8 ET A UNICK8 +46 (PHYSIO)  ***

debhb0: movem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        aos cpthb1(2)

        move 2,saveac
        RET


;*** MESURE DU TEST DU 'TIMER' ET POSITIONNEMENT,INSERER A UNIOVO +3 ***
;                                                       (PHYSIO)     ***

tstip0: dmovem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        skipn debupo(2)
        jrst tstip9

        move 3,@todclk
        sub 3,debupo(2)
        addm 3,moypos(2)
        imul 3,3
        addm 3,varpos(2)
        aos cptpos(2)
tstip9: dmove 2,saveac
        RET
```

```
;*** MESURE DU TEST DU 'TIMER' ET TRANSFERT,INSERER A UNIOVO +5 ***
;                                                          (PHYSIO)***

tstit0: dmovem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        skipn debutr(2)
        jrst tstit9

        move 3,@todclk
        sub 3,debutr(2)
        addm 3,moytra(2)
        imul 3,3
        addm 3,vartra(2)
        aos cpttra(2)

tstit9: dmove 2,saveac
        RET


;*** MESURE DU NBRE D'IORBs AJOUTES DS TWQ LORS DE  L'ELIMINATION D'UN IORB,
;                                        INSERER A DONIRB +16 (PHYSIO) ***


elirb0: movem 2,saveac

        ldb 2,ptuni             ;saisie du no d'unite inscrit ds UDBSTS avant

        aos cptirb(2)

        move 2,saveac
        RET


endmon=.-1              ; fin des routines a inserer dans le moniteur
```

```
        reloc 5000

;*****************************************
;***  PARTIE UTILISATEUR DU PROGRAMME ***
;*****************************************

pdlen==10
stack: block pdlen

moytty: 0       ; nbre moyen de processus aux TTY's
ttyfil: block 6 ; nbre moyen de processus aux TTY's par file
mohal1: 0       ; high-queue load average
molal1: 0       ; low-queue load average
nbbls: 0        ; nbre de proc dans le balance set
cpu%: 0         ; %tages d'utilisation du cpu
uni%: block 5   ; %tages d'utilisation des unites
can%: block 4   ; %tages d'utilisation des canaux
cptuni: 0       ; nbre total de mesures sortant du cpu vers les unites

tpsmes: 0       ; temps de la mesure en millisecondes

arocent: ^d100          ;

errtrt:
        CALL error

        LPRINT <>
        LPRINT <??? PERFO ??? FATAL ERROR, CAN'T CONTINUE>
        LPRINT <>
        JSY haltf,-,-,-,-,no
        jrst .-1

                                        ;*** PROGRAMME PRINCIPAL ***

start:  JSY reset                       ;*** INITIALISATIONS ***

        move p,[iowd pdlen,stack]

        move 6,[070312,,0]
        movem 6,ptuni
        move 6,[220302,,0]
        movem 6,ptfil
        move 6,[SIXBIT/PS/]
        movem 6,ps
        move 6,[SIXBIT/US/]
        movem 6,us



        CALL lock
        CALL mreloc
        CALL defbkpt
        CALL addmon
        CALL minser

        fltr 6,arocent
        movem 6,arocent

        JSY time
        movem 1,tpsmes
```

```
                        ;***1ere MESURE DU NBRE DE PROC DS BALSET ***
hrri 1,.systat
hrli 1,12
JSY getab
movem 1,nbbls

FOR 6,1,^d720,1         ;*** DISMISS ET MESURE DU LOAD AVERAGE ***
        JSY dismss,^d5000
        JSY getab,%I(.halav)
        fadm 1,mohall
        JSY getab,%I(.lalav)
        fadm 1,molall
OD

CALL mrembp
CALL munloc

JSY time
subm 1,tpsmes
fltr 6,tpsmes
movem 6,tpsmes

                        ;***2ieme MESURE DU NBRE DE PROC DS BALSET ***

hrri 1,.systat
hrli 1,12
JSY.getab
sub 1,nbbls
fltr 6,1
fdv 6,tpsmes
movem 6,nbbls
```

```
;*** CALCUL ET IMPRESSION DES RESULTATS ***


move 6,mohal1
fdvri 6,(720.0)
movem 6,mohal1
move 6,molal1
fdvri 6,(720.0)
movem 6,molal1

LPRINT <>
LPRINT <  LOAD AVERAGE  >
LPRINT <----------->
LPRINT <    HIGH-QUEUE :  >
JSY flout,%I(.priou),mohal1,[24037,,30200]
LPRINT <    LOW-QUEUE :  >
JSY flout,%I(.priou),molal1,[24037,,30200]
LPRINT <>

setz 6,
FOR 5,0,5,1
        add 6,moyter(5)
OD
fltr 4,6
fdv 4,tpsmes
movem 4,moytty

FOR 5,0,5,1
        fltr 6,moyter(5)
        fdv 6,tpsmes
        movem 6,ttyfil(5)
OD

LPRINT <>
LPRINT <  NOMBRE DE PROCESSUS AUX TERMINAUX  >
LPRINT <-------------------------->
LPRINT <    MOYENNE GLOBALE : >
JSY flout,%I(.priou),moytty,[24037,,30200]
LPRINT <    MOYENNE          FIL0       FIL1       FIL2       FIL3       FIL4       FIL5>
LPRINT <                    >
FOR 5,0,5,1
        JSY flout,%I(.priou),ttyfil(5),[24037,,30200]
        PRINT <      >
OD
LPRINT <>

LPRINT <>
LPRINT <  NOMBRE DE PROCESSUS DANS LE BALANCE SET  >
LPRINT <-------------------------->
LPRINT <        MOYENNE : >
JSY flout,%I(.priou),nbbls,[24037,,30200]
LPRINT <>


setz 6,                         ;calcul utilisation CPU
FOR 5,0,5,1
        fltr 4,moycpu(5)
        fad 6,4
OD
fdv 6,tpsmes
fmpr 6,arocent
movem 6,cpu%

FOR 5,0,4,1                     ;calcul utilisation unites
        fltr 6,moypos(5)
        fltr 4,moytra(5)
        fad 6,4
        fdv 6,tpsmes
        fmpr 6,arocent
        movem 6,uni%(5)
OD
```

```
fltr 6,moytra                    !calcul utilisation canaux
fltr 4,moytra+3
fad 6,4
fdv 6,tpsmes
fmpr 6,arocent
movem 6,can%

fltr 6,moytra+1
fltr 4,moytra+4
fad 6,4
fdv 6,tpsmes
fmpr 6,arocent
movem 6,can%+1

fltr 6,moytra+2
fdv 6,tpsmes
fmpr 6,arocent
movem 6,can%+2

fltr 6,moytra+5
fdv 6,tpsmes
fmpr 6,arocent
movem 6,can%+3

LPRINT <>
LPRINT <  POURCENTAGE D'UTILISATION DES SERVEURS  >
LPRINT <---------------------------------------->
LPRINT <>
LPRINT <    CPU : >
JSY flout,%I(.priou),cpu%,[24037,,30200]
LPRINT <>
LPRINT <    UNITES :  PS1   PS2   PS3   US    MONT>
LPRINT <             >
FOR 5,0,4,1
        JSY flout,%I(.priou),uni%(5),[24037,,30200]
OD
LPRINT <>
LPRINT <    CANAUX :    0     1     2     3>
LPRINT <               >
FOR 5,0,3,1
        JSY flout,%I(.priou),can%(5),[24037,,30200]
OD
LPRINT <>


fix 6,tpsmes
movem 6,tpsmes

LPRINT <>
LPRINT <  TEMPS D'EXECUTION DU MESUREUR : >
JSY nout,%I(.priou),tpsmes,[13,,12]
LPRINT <>

FOR 5,0,5,1
        move 6,moyter(5)
        idiv 6,cptter(5)
        movem 6,moyter(5)
OD

LPRINT <>
LPRINT <  TEMPS PASSE AU TERMINAL (PAR FILE)  >
LPRINT <-------------------------------------->
LPRINT <         >
LPRINT <                      FIL0       FIL1       FIL2       FIL3       FIL4       FIL5 >
LPRINT <>
LPRINT <  NBRE DE MESURES : >
FOR 5,0,5,1
        JSY nout,%I(.priou),cptter(5),[13,,12]
OD
LPRINT <  MOYENNE :          >
FOR 5,0,5,1
        JSY nout,%I(.priou),moyter(5),[13,,12]
OD
LPRINT <>
```

```
FOR 5,0,5,1

        move 6,moycpu(5)
        idiv 6,cptcpo(5)
        movem 6,moycpu(5)
        imul 6,6  .
        move 3,varcpu(5)
        idiv 3,cptcpo(5)
        sub 3,6
        movem 3,varcpu(5)
OD

LPRINT <>
LPRINT <  TEMPS PASSE DANS LE CPU (PAR FILE)  >
LPRINT <------------------------------------->
LPRINT <>
LPRINT <  NBRE D'ENTREES : >
JSY nout,XI(.priou),cptcpi,^d10
LPRINT <>
LPRINT <                    FIL0       FIL1       FIL2       FIL3       FIL4       FIL5 >
LPRINT <>
LPRINT <  NBRE DE SORTIES : >
FOR 5,0,5,1
        JSY nout,XI(.priou),cptcpo(5),[13,,12]
OD
LPRINT <  MOYENNE :          >
FOR 5,0,5,1
        JSY nout,XI(.priou),moycpu(5),[13,,12]
OD
LPRINT <  VARIANCE :         >
FOR 5,0,5,1
        JSY nout,XI(.priou),varcpu(5),[13,,12]
OD
LPRINT <>

                ; calcul des prob de trans sous forme de %
FOR 5,0,5,1
        fltr 6,cptcpo(5)
        movem 6,cptcpo(5)

        fltr 6,outter(5)
        fdv 6,cptcpo(5)
        fmpr 6,arocent
        movem 6,outter(5)

        fltr 6,outcpu(5)
        fdv 6,cptcpo(5)
        fmpr 6,arocent
        movem 6,outcpu(5)

        fltr 6,outsol(5)
        fdv 6,cptcpo(5)
        fmpr 6,arocent
        movem 6,outsol(5)

        fltr 6,outall(5)
        fdv 6,cptcpo(5)
        fmpr 6,arocent
        movem 6,outall(5)

OD

LPRINT <>
LPRINT <  PROBABILITES DE TRANSITION  >
LPRINT <---------------------------->
LPRINT <>
LPRINT <                    FIL0       FIL1       FIL2       FIL3       FIL4       FIL5 >
LPRINT <>
LPRINT <  %CPU-)TERMINAUX :>
FOR 5,0,5,1
        JSY flout,XI(.priou),outter(5),[24037,,30200]
        PRINT <      >
OD
LPRINT <  %CPU-)CPU :        >
FOR 5,0,5,1
        JSY flout,XI(.priou),outcpu(5),[24037,,30200]
        PRINT <      >
OD
LPRINT <  %CPU-)SOLISTE :  >
FOR 5,0,5,1
        JSY flout,XI(.priou),outsol(5),[24037,,30200]
        PRINT <      >
OD
LPRINT <  %CPU-)UNITES :   >
FOR 5,0,5,1
        JSY flout,XI(.priou),outall(5),[24037,,30200]
        PRINT <      >
OD
LPRINT <>
```

```
FOR 5,0,5,1
        fltr 6,outuni(5)
        fadm 6,cptuni
OD

FOR 5,0,5,1
        fltr 6,outuni(5)
        fdv 6,cptuni
        fmpr 6,arocent
        movem 6,outuni(5)
OD

LPRINT <>
LPRINT <  POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE  >
LPRINT <-------------------------------------------------------->
LPRINT <>
LPRINT <    PS:1 : >
JSY flout,%I(.priou),outuni,[24037,,30200]
LPRINT <    PS:2 : >
JSY flout,%I(.priou),outuni+1,[24037,,30200]
LPRINT <    PS:3 : >
JSY flout,%I(.priou),outuni+2,[24037,,30200]
LPRINT <    US : >
JSY flout,%I(.priou),outuni+3,[24037,,30200]
LPRINT <    DSK MONTABLE : >
JSY flout,%I(.priou),outuni+4,[24037,,30200]
LPRINT <    MTA : >
JSY flout,%I(.priou),outuni+5,[24037,,30200]
LPRINT <>


FOR 5,0,4,1
        move 6,moypos(5)
        idiv 6,cptpos(5)
        movem 6,moypos(5)
        imul 6,6
        move 3,varpos(5)
        idiv 3,cptpos(5)
        sub 3,6
        movem 3,varpos(5)
OD

LPRINT <>
LPRINT <  TEMPS DE POSITIONNEMENT  >
LPRINT <---------------------------->
LPRINT <>
LPRINT <                    PS1         PS2         PS3         US          MONT>
LPRINT <>
LPRINT <    NBR DE MCS : >
FOR 5,0,4,1
        JSY nout,%I(.priou),cptpos(5),[12,,12]
        PRINT < >
OD
LPRINT <     MOYENNE :    >
FOR 5,0,4,1
        JSY nout,%I(.priou),moypos(5),[12,,12]
        PRINT < >
OD
LPRINT <     VARIANCE :   >
FOR 5,0,4,1
        JSY nout,%I(.priou),varpos(5),[12,,12]
        PRINT < >
OD
LPRINT <>

FOR 5,0,4,1
        fltr 4,cpscan(5)
        fltr 6,cptpos(5)
        fdv 4,6
        movem 4,cpscan(5)
OD

LPRINT <>
LPRINT <  NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ  >
LPRINT <-------------------------------------------------------------------------------->
LPRINT <>
LPRINT <                    PS1         PS2         PS3         US          MONT>
LPRINT <>
LPRINT <                   >
FOR 5,0,4,1
        JSY flout,%I(.priou),cpscan(5),[24037,,30200]
        PRINT <      >
OD
```

```
move 6,cpttra+3
addm 6,cpttra
move 6,cpttra+4
addm 6,cpttra+1
move 6,cpttra+5
movem 6,cpttra+3

move 6,moytra+3
addm 6,moytra
move 6,moytra+4
addm 6,moytra+1
move 6,moytra+5
movem 6,moytra+3

move 6,vartra+3
addm 6,vartra
move 6,vartra+4
addm 6,vartra+1
move 6,vartra+5
movem 6,vartra+3

FOR 5,0,3,1
        move 6,moytra(5)
        idiv 6,cpttra(5)
        movem 6,moytra(5)
        imul 6,6
        move 3,vartra(5)
        idiv 3,cpttra(5)
        sub 3,6
        movem 3,vartra(5)
OD

LPRINT <>
LPRINT <  TEMPS DE TRANSFERT  >
LPRINT <-------------------->
LPRINT <>
LPRINT <    CANAUX          0          1          2          3>
LPRINT <>
LPRINT <     NBR DE MES :   >
FOR 5,0,3,1
        JSY nout,%I(.priou),cpttra(5),[12,,12]
        PRINT < >
OD
LPRINT <     MOYENNE :      >
FOR 5,0,3,1
        JSY nout,%I(.priou),moytra(5),[12,,12]
        PRINT < >
OD
LPRINT <     VARIANCE :     >
FOR 5,0,3,1
        JSY nout,%I(.priou),vartra(5),[12,,12]
        PRINT < >
OD
LPRINT <>

LPRINT <>
LPRINT <  NOMBRE DE VERIFICATIONS DU 'HOME BLOCK'  >
LPRINT <-------------------------------------------->
LPRINT <                              (AJOUT D'UN IORB EN TETE DE TWQ)>
LPRINT <>
LPRINT <                    PS1        PS2        PS3        US          MONT        MTA>
LPRINT <>
LPRINT <                    >
FOR 5,0,5,1
        JSY nout,%I(.priou),cpthbl(5),[12,,12]
        PRINT < >
OD
LPRINT <>

LPRINT <>
LPRINT <  NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE  >
LPRINT <------------------------------------------------------------>
LPRINT <>
LPRINT <                    PS1        PS2        PS3        US          MONT        MTA>
LPRINT <>
LPRINT <                    >
FOR 5,0,5,1
        JSY nout,%I(.priou),abortr(5),[12,,12]
        PRINT < >
OD
LPRINT <>
```

```
        LPRINT <>
        LPRINT <  NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB  >
        LPRINT <------------------------------------------------------------->
        LPRINT <>
        LPRINT <                    PS1         PS2         PS3         US          MONT        MTA>
        LPRINT <>
        LPRINT <                    >
        FOR 5,0,5,1
                JSY nout,XI(.priou),cptirb(5),[12,,12]
                PRINT < >
        OD
        LPRINT <>

FIN start
```

A N N E X E F

LES MESURES OBTENUES

LOAD AVERAGE

HIGH-QUEUE : 2.57
LOW-QUEUE : 2.27

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 23.51

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0.00	17.87	3.85	1.46	0.31	0.00

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 4.85

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 44.46

UNITES :	PS1	PS2	PS3	US	MONT
	27.04	27.40	25.10	2.91	0.04

CANAUX :	0	1	2	3
	20.47	18.01	17.42	37.22

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	35292	5165	2932	54	22
MOYENNE :	0	1842	2714	1820	20993	120

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 235800

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	1480	55979	30210	56010	36248	4289
MOYENNE :	5	8	7	8	10	18
VARIANCE :	98	129	153	161	146	448

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0.00	63.23	16.82	5.21	0.15	0.51
%CPU-)CPU :	0.68	0.59	0.56	0.24	0.54	1.40
%CPU-)GOLISTE :	0.00	0.03	0.06	0.10	0.17	0.98
%CPU-)UNITES :	99.32	36.15	82.56	94.45	99.14	97.11

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 26.74
PS:2 : 27.55
PS:3 : 24.15
US : 3.61
DSK MONTABLE : 0.06
MTA : 17.90

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	16981	18815	14046	1865	37
MOYENNE :	18	18	19	18	10
VARIANCE :	176	158	199	116	52

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2.20	2.04	2.40	2.71	2.22

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	42383	38553	33723	24867
MOYENNE :	17	17	18	54
VARIANCE :	523	372	504	13493

NOMBRE DE VERIFICATIONS DU "HOME BLOCK"

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 3.36
LOW-QUEUE : 2.82

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 24.20

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0.00	17.21	4.72	2.07	0.15	0.03

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 6.07

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 52.45

UNITES :	PS1	PS2	PS3	US	MONT
	32.92	26.08	26.76	13.56	0.01

CANAUX :	0	1	2	3
	32.38	18.25	18.67	6.14

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	37271	5746	4406	46	722
MOYENNE :	0	1680	2989	1713	12474	152

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 253082

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	1373	67759	42600	41371	34678	9249
MOYENNE :	4	8	7	10	13	14
VARIANCE :	97	126	155	142	257	262

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0.00	55.22	13.17	10.61	0.13	7.81
%CPU-)CPU :	0.87	0.69	0.65	0.27	0.56	0.29
%CPU-)GOLISTE :	0.00	0.01	0.07	0.13	0.08	0.31
%CPU-)UNITES :	99.13	44.08	86.11	88.98	99.23	91.59

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 32.27
PS:2 : 25.57
PS:3 : 25.41
US : 15.89
DSK MONTABLE : 0.01
MTA : 0.85

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	23795	19069	17361	5405	8
MOYENNE :	16	15	17	21	11
VARIANCE :	201	130	154	170	61

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2.00	1.98	2.16	4.34	2.25

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	71077	37766	37510	1223
MOYENNE :	16	17	18	183
VARIANCE :	391	689	628	2406

NOMBRE DE VERIFICATIONS DU "HOME BLOCK"

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 1,90
LOW-QUEUE : 2,56

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 19,45

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0,00	15,00	3,61	0,59	0,24	0,00

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 4,43

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 45,95

UNITES :	PS1	PS2	PS3	US	MONT
	21,00	21,09	20,40	4,88	0,00

CANAUX :	0	1	2	3
	17,89	14,78	15,34	5,60

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	29833	3694	843	41	1
MOYENNE :	0	1829	3555	2577	21466	92

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 204939

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	1244	45832	22525	21585	44937	3610
MOYENNE :	8	10	10	14	12	19
VARIANCE :	126	184	226	325	207	369

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0.00	65.26	16.10	3.84	0.09	0.03
%CPU-)CPU :	1.21	0.75	0.73	0.50	0.75	1.11
%CPU-)GOLISTE :	0.00	0.01	0.13	0.25	0.13	0.69
%CPU-)UNITES :	98.79	33.98	83.03	95.41	99.03	98.17

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 32,33
PS:2 : 30,44
PS:3 : 29,19
US : 7,04
DSK MONTABLE : 0,00
MTA : 1,00

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	13205	12508	10371	3662	0
MOYENNE :	16	18	17	22	0
VARIANCE :	136	180	186	195	0

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2,55	2,53	2,93	2,00	0,00

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	41012	31700	30396	1034
MOYENNE :	16	17	18	199
VARIANCE :	432	526	637	8813

NOMBRE DE VERIFICATIONS DU 'HOME BLOCK'

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 2,20
LOW-QUEUE : 1,93

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 28,02

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0,00	20,58	6,70	0,52	0,19	0,00

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 4,16

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 43,51

UNITES :	PS1	PS2	PS3	US	MONT
	24,44	17,11	21,88	4,74	0,03

CANAUX :	0	1	2	3
	20,59	12,41	14,47	7,43

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	43562	5657	896	86	129
MOYENNE :	0	1718	4309	2144	8052	572

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 219813

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	7083	68762	34819	14260	11341	22961
MOYENNE :	7	7	8	14	17	13
VARIANCE :	67	99	153	430	400	459

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0,00	63,54	15,91	6,18	0,75	0,56
%CPU-)CPU :	0,42	0,54	0,53	0,57	1,02	0,48
%CPU-)GOLISTE :	0,00	0,01	0,04	0,17	0,17	0,14
%CPU-)UNITES :	99,58	35,92	83,52	93,09	98,06	98,82

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 35,38
PS:2 : 22,85
PS:3 : 29,34
US : 7,33
DSK MONTABLE : 0,06
MTA : 5,04

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	18353	11398	15551	1811	7
MOYENNE :	15	15	17	20	19
VARIANCE :	135	154	150	141	124

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2,08	2,16	2,04	4,37	9,29

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	46092	24722	31663	5427
MOYENNE :	16	18	16	50
VARIANCE :	449	813	539	3079

NOMBRE DE VERIFICATIONS DU 'HOME BLOCK'

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 1,39
LOW-QUEUE : 2,44

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 39,95

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0,00	15,10	23,14	1,51	0,17	0,00

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 3,84

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 42,54

UNITES :	PS1	PS2	PS3	US	MONT
	22,85	21,21	19,32	1,87	0,01

CANAUX :	0	1	2	3
	17,53	16,72	14,42	8,92

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	33861	4951	2033	20	93
MOYENNE :	0	1622	17002	2717	32309	149

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 197165

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	1000	54809	29122	12525	8954	23456
MOYENNE :	5	8	8	11	23	20
VARIANCE :	73	152	181	212	452	564

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0,00	61,98	16,68	16,13	0,21	0,40
%CPU-)CPU :	0,80	0,71	0,58	0,17	0,51	0,70
%CPU-)GOLISTE :	0,00	0,01	0,05	0,15	0,06	0,24
%CPU-)UNITES :	99,20	37,30	82,68	83,55	99,22	98,66

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 35,42
PS:2 : 29,58
PS:3 : 30,05
US : 2,91
DSK MONTABLE : 0,03
MTA : 2,01

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	14236	10895	10581	1301	12
MOYENNE :	16	14	16	23	5
VARIANCE :	142	172	182	268	22

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2,19	2,39	2,50	1,97	2,25

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	33732	26061	26444	1760
MOYENNE :	18	23	19	183
VARIANCE :	1015	1397	1053	254

NOMBRE DE VERIFICATIONS DU 'HOME BLOCK'

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 2,54
LOW-QUEUE : 2,20

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 24,89

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0,00	19,84	4,11	0,61	0,31	0,00

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 4,73

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 46,53

UNITES :	PS1	PS2	PS3	US	MONT
	28,08	26,93	28,33	8,40	0,00

CANAUX :	0	1	2	3
	24,55	18,27	19,27	11,98

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	36072	6014	1485	34	100
MOYENNE :	0	2000	2490	1513	33158	149

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 245293

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	5939	58369	35689	32600	31044	25153
MOYENNE :	8	7	6	9	12	11
VARIANCE :	120	102	119	147	266	246

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU->TERMINAUX :	0.00	62.01	16.58	4.48	0.11	0.40
%CPU->CPU :	0.72	0.54	0.64	0.27	0.57	0.25
%CPU->GOLISTE :	0.02	0.01	0.10	0.17	0.11	0.23
%CPU->UNITES :	99.26	37.44	82.68	95.08	99.20	99.13

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 30.59
PS:2 : 27.37
PS:3 : 28.85
US : 9.77
DSK MONTABLE : 0.01
MTA : 3.42

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	20506	17411	17841	4373	4
MOYENNE :	16	18	18	19	9
VARIANCE :	153	155	144	175	29

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2.15	2.26	2.33	3.22	2.25

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	58111	39414	41533	4891
MOYENNE :	15	16	16	89
VARIANCE :	297	445	395	10998

NOMBRE DE VERIFICATIONS DU "HOME BLOCK"

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 1.96
LOW-QUEUE : 2.53

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 22.87

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0.00	16.99	4.62	1.05	0.18	0.00

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 4.48

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 46.70

UNITES :	PS1	PS2	PS3	US	MONT
	21.71	17.40	19.92	6.21	0.35

CANAUX :	0	1	2	3
	19.02	12.44	14.19	6.70

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	1	32170	5304	983	34	72
MOYENNE :	3624	1922	3169	3899	20026	143

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 203622

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	1198	53892	34019	21493	16731	19449
MOYENNE :	5	10	9	8	15	17
VARIANCE :	66	169	248	186	538	473

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0.08	59.97	15.21	4.50	0.20	0.37
%CPU-)CPU :	0.67	0.99	0.53	0.28	0.71	0.50
%CPU-)GOLISTE :	0.00	0.01	0.08	0.17	0.10	0.21
%CPU-)UNITES :	99.25	39.03	84.19	95.05	98.98	98.92

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 30.47
PS:2 : 24.54
PS:3 : 26.65
US : 9.61
DSK MONTABLE : 0.34
MTA : 8.39

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	14803	12256	12879	4387	27
MOYENNE :	15	15	16	20	15
VARIANCE :	156	165	152	161	119

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2.20	2.14	2.22	2.35	13.41

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	42924	26640	28541	7788
MOYENNE :	16	16	18	31
VARIANCE :	395	528	596	116

NOMBRE DE VERIFICATIONS DU "HOME BLOCK"

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

LOAD AVERAGE

HIGH-QUEUE : 1.61
LOW-QUEUE : 2.87

NOMBRE DE PROCESSUS AUX TERMINAUX

MOYENNE GLOBALE : 21.57

MOYENNE	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
	0.00	14.41	5.98	1.10	0.07	0.0

0

NOMBRE DE PROCESSUS DANS LE BALANCE SET

MOYENNE : 4.48

POURCENTAGE D'UTILISATION DES SERVEURS

CPU : 34.08

UNITES :	PS1	PS2	PS3	US	MONT
	20.10	16.75	17.34	0.65	0.01

CANAUX :	0	1	2	3
	14.58	12.50	12.29	4.83

TEMPS PASSE AU TERMINAL (PAR FILE)

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE MESURES :	0	27714	4469	2089	47	8
MOYENNE :	0	1914	4929	1944	5566	926

TEMPS PASSE DANS LE CPU (PAR FILE)

NBRE D'ENTREES : 193902

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
NBRE DE SORTIES :	1342	42332	25457	21914	29423	2534
MOYENNE :	6	7	7	9	15	20
VARIANCE :	69	90	197	258	477	543

PROBABILITES DE TRANSITION

	FIL0	FIL1	FIL2	FIL3	FIL4	FIL5
%CPU-)TERMINAUX :	0.00	65.64	17.35	9.46	0.15	0.32
%CPU-)CPU :	1.19	0.54	0.69	0.30	1.90	0.87
%CPU-)GOLISTE :	0.00	0.00	0.07	0.19	0.13	0.99
%CPU-)UNITES :	98.81	33.82	81.88	90.05	97.82	97.83

POURCENTAGE POUR CHAQUE UNITE QUELLE QUE SOIT LA FILE

PS:1 : 33.15
PS:2 : 27.44
PS:3 : 28.84
US : 1.35
DSK MONTABLE : 0.02
MTA : 9.19

TEMPS DE POSITIONNEMENT

	PS1	PS2	PS3	US	MONT
NBR DE MES :	12111	9522	10749	521	6
MOYENNE :	17	16	17	19	11
VARIANCE :	178	152	142	174	18

NOMBRE MOYEN,POUR UN POSITIONNEMENT, D'IORBs TRANSFERES DE LA PWQ VERS LA TWQ

PS1	PS2	PS3	US	MONT
2.39	2.52	2.35	2.26	2.67

TEMPS DE TRANSFERT

CANAUX	0	1	2	3
NBR DE MES :	30181	24025	25234	3290
MOYENNE :	17	19	17	54
VARIANCE :	513	564	586	3952

NOMBRE DE VERIFICATIONS DU "HOME BLOCK"

(AJOUT D'UN IORB EN TETE DE TWQ)

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NOMBRE DE SUPPRESSIONS DE TTES LES DEMANDES D'UNE UNITE

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

NBRE D'IORBs AJOUTES DS TWQ LORS DE L'ELIMINATION D'UN IORB

PS1	PS2	PS3	US	MONT	MTA
0	0	0	0	0	0

A N N E X E G

LE MODELE PROGRAMME EN QNAP

```
/CONTROL / OPTION=NSOURCE;


/DECLARE / QUEUE TTY,SEMMEM,WAITMEM,LIBTTY,LIBFIL,CPU;
           QUEUE PS1,PS2,PS3,US,MONT,CANAL0,CANAL1,CANAL2,CANAL3;

           INTEGER INI,NBMEM,CPT(5),CHANGT(5);

           REAL SERTTY,SERCPU,PROBIN(6);
           REAL PRTTY,PRCPU,PRFIL,PRUNI;
           REAL POSIT(5),TRANSF(4);

           BOOLEAN PREM1,PREM2;

           CUSTOMER INTEGER UNIT;

           REF CUSTOMER CLIENT1,CLIENT2,CLIENT3,CLIENT4,CLIENT5;

           FLAG SYNC1,SYNC2,SYNC3,SYNC4,SYNC5,SYNTR1,SYNTR2;
```

```
/STATION/ NAME=TTY;
          TYPE=INFINITE;
          SCHED=FIFO;
          SERVICE=EXP(SERTTY);
          TRANSIT=WAITMEM;


/STATION/ NAME=SEMMEM;
          TYPE=SEMAPHORE,MULTIPLE(NBMEM);
          SCHED=FIFO;


/STATION/ NAME=WAITMEM;
          TYPE=INFINITE;
          INIT=INI;
          SERVICE=P(SEMMEM);
          TRANSIT=CPU;

/STATION/ NAME=LIBTTY;
          TYPE=INFINITE;
          SERVICE=V(SEMMEM);
          TRANSIT=TTY;

/STATION/ NAME=LIBFIL;
          TYPE=INFINITE;
          SERVICE=V(SEMMEM);
          TRANSIT=WAITMEM;


/STATION/ NAME=CPU;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=EXP(SERCPU);
          TRANSIT=LIBTTY,PRTTY,
                  CPU,PRCPU,
                  LIBFIL,PRFIL,
                  PS1,PROBIN(1),
                  PS2,PROBIN(2),
                  PS3,PROBIN(3),
                  US,PROBIN(4),
                  MONT,PROBIN(5),
                  CANAL3,PROBIN(6);


/STATION/ NAME=PS1;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                       IF PREM1=TRUE THEN PREM1:=FALSE
                                     ELSE WAIT(SYNTR1);
                       RESET(SYNTR1);
                       EXP(POSIT(1));
                       FOR CPT(1):=1 STEP 1 UNTIL CHANGT(1)
                         DO BEGIN
                               CLIENT1:=CUSTOMER.NEXT;
                               IF CLIENT1 <> NIL
                                 THEN BEGIN
                                           TRANSIT(CLIENT1,CANAL0);
                                           CLIENT1.UNIT:=1;
                                      END;
                            END;
                       CLIENT1:=NEW(CUSTOMER);
                       CLIENT1.UNIT:=91;
                       TRANSIT(CLIENT1,CANAL0);
                       SET(SYNTR1);
                       RESET(SYNC1);
                       WAIT(SYNC1);
                  END;
          TRANSIT=OUT;
```

```
/STATION/ NAME=PS2;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                        IF PREM2=TRUE THEN PREM2:=FALSE
                                      ELSE WAIT(SYNTR2);
                        RESET(SYNTR2);
                        EXP(POSIT(2));
                        FOR CPT(2):=1 STEP 1 UNTIL CHANGT(2)
                          DO BEGIN
                                CLIENT2:=CUSTOMER.NEXT;
                                IF CLIENT2 <> NIL
                                  THEN BEGIN
                                              TRANSIT(CLIENT2,CANAL1);
                                              CLIENT2.UNIT:=2;
                                       END;
                             END;
                        CLIENT2:=NEW(CUSTOMER);
                        CLIENT2.UNIT:=92;
                        TRANSIT(CLIENT2,CANAL1);
                        SET(SYNTR2);
                        RESET(SYNC2);
                        WAIT(SYNC2);
                  END;
          TRANSIT=OUT;


/STATION/ NAME=PS3;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                       EXP(POSIT(3));
                       FOR CPT(3):=1 STEP 1 UNTIL CHANGT(3)
                         DO BEGIN
                               CLIENT3:=CUSTOMER.NEXT;
                               IF CLIENT3 <> NIL
                                 THEN BEGIN
                                             TRANSIT(CLIENT3,CANAL2);
                                             CLIENT3.UNIT:=3;
                                      END;
                            END;
                         CLIENT3:=NEW(CUSTOMER);
                         CLIENT3.UNIT:=93;
                         TRANSIT(CLIENT3,CANAL2);
                         RESET(SYNC3);
                         WAIT(SYNC3);
                   END;
           TRANSIT=OUT;

 /STATION/ NAME=US;
           TYPE=SINGLE;
           SCHED=FIFO;
           SERVICE=BEGIN
                        IF PREM1=TRUE THEN PREM1:=FALSE
                                      ELSE WAIT(SYNTR1);
                        RESET(SYNTR1);
                        EXP(POSIT(4));
                        FOR CPT(4):=1 STEP 1 UNTIL CHANGT(4)
                          DO BEGIN
                                CLIENT4:=CUSTOMER.NEXT;
                                IF CLIENT4 <> NIL
                                  THEN BEGIN
                                              TRANSIT(CLIENT4,CANAL0);
                                              CLIENT4.UNIT:=4;
                                       END;
                             END;
                        CLIENT4:=NEW(CUSTOMER);
                        CLIENT4.UNIT:=94;
                        TRANSIT(CLIENT4,CANAL0);
                        SET(SYNTR1);
                        RESET(SYNC4);
                        WAIT(SYNC4);
                  END;
           TRANSIT=OUT;
```

```
/STATION/ NAME=MONT;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                        IF PREM2=TRUE THEN PREM2:=FALSE
                                      ELSE WAIT(SYNTR2);
                        RESET(SYNTR2);
                        EXP(POSIT(5));
                        FOR CPT(5):=1 STEP 1 UNTIL CHANGT(5)
                          DO BEGIN
                                CLIENT5:=CUSTOMER.NEXT;
                                IF CLIENT5 <> NIL
                                  THEN BEGIN
                                             TRANSIT(CLIENT5,CANAL1);
                                             CLIENT5.UNIT:=5;
                                       END;
                             END;
                        CLIENT5:=NEW(CUSTOMER);
                        CLIENT5.UNIT:=95;
                        TRANSIT(CLIENT5,CANAL1);
                        SET(SYNTR2);
                        RESET(SYNC5);
                        WAIT(SYNC5);
                   END;
          TRANSIT=OUT;


/STATION/ NAME=CANAL0;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                        EXP(TRANSF(1));
                        IF CUSTOMER.UNIT=91
                          THEN SET(SYNC1)
                          ELSE IF CUSTOMER.UNIT=94
                                 THEN SET(SYNC4);
                   END;
          TRANSIT=CPU;


/STATION/ NAME=CANAL1;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                        EXP(TRANSF(2));
                        IF CUSTOMER.UNIT=92
                          THEN SET(SYNC2)
                                    ELSE IF CUSTOMER.UNIT=95
                                           THEN SET(SYNC5);
                   END;
          TRANSIT=CPU;


/STATION/ NAME=CANAL2;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=BEGIN
                        EXP(TRANSF(3));
                        IF CUSTOMER.UNIT=93
                          THEN SET(SYNC3);
                   END;
          TRANSIT=CPU;


/STATION/ NAME=CANAL3;
          TYPE=SINGLE;
          SCHED=FIFO;
          SERVICE=EXP(TRANSF(4));
          TRANSIT=CPU;
```

```
/CONTROL/ ACCURACY=CPU,TTY,PS1,PS2,PS3,US,MONT,CANAL0,CANAL1,CANAL2,CANAL3;
          TMAX=1000000.;


/EXEC/ BEGIN

        PREM1:=TRUE;
        PREM2:=TRUE;


        PRINT('*************************************************************');
        PRINT('*                                                           *');
        PRINT('*   CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP      *');
        PRINT('*              AVEC LES PARAMETRES SUIVANTS                 *');
        PRINT('*                                                           *');
        PRINT('*************************************************************');
        PRINT(' ');
        PRINT(' ');
        PRINT(' ');
        INI:=GET(INTEGER);
        PRINT(' NOMBRE DE PROCESSUS DANS LE MODELE : ',INI);
        NBMEM:=GET(INTEGER);
        PRINT(' NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET : ',NBMEM);
        SERTTY:=GET(REAL);
        PRINTF('(1H ,' TEMPS MOYEN DE SERVICE DES TERMINAUX : ',F12.4)',SERTTY);
        SERCPU:=GET(REAL);
        PRINTF('(1H ,' TEMPS MOYEN DE SERVICE DU CPU : ',F12.6)',SERCPU);
        PRTTY:=GET(REAL);
        PRINT(' PROBABILITES DE TRANSITION A LA SORTIE DU CPU ');
        PRINTF('(1H ,'          cpu--->terminaux : ',F6.3)',PRTTY);
        PRCPU:=GET(REAL);
        PRINTF('(1H ,'          cpu--->cpu : ',F6.3)',PRCPU);
        PRFIL:=GET(REAL);
        PRINTF('(1H ,'          cpu--->entree dans balset : ',F6.3)',PRFIL);
        PRUNI:=GET(REAL);
        PRINTF('(1H ,'          cpu--->unites : ',F6.3)',PRUNI);
        PROBIN(1):=GET(REAL);
        PRINT(' POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT ');
        PRINTF('(1H ,'          ps:1 : ',F5.2)',PROBIN(1));
        PROBIN(2):=GET(REAL);
        PRINTF('(1H ,'          ps:2 : ',F5.2)',PROBIN(2));
        PROBIN(3):=GET(REAL);
        PRINTF('(1H ,'          ps:3 : ',F5.2)',PROBIN(3));
        PROBIN(4):=GET(REAL);
        PRINTF('(1H ,'          us : ',F5.2)',PROBIN(4));
        PROBIN(5):=GET(REAL);
        PRINTF('(1H ,'          montable : ',F5.2)',PROBIN(5));
        PROBIN(6):=GET(REAL);
        PRINTF('(1H ,'          bandes : ',F5.2)',PROBIN(6));

        PROBIN(1):=(PROBIN(1)*PRUNI)/100.;
        PROBIN(2):=(PROBIN(2)*PRUNI)/100.;
        PROBIN(3):=(PROBIN(3)*PRUNI)/100.;
        PROBIN(4):=(PROBIN(4)*PRUNI)/100.;
        PROBIN(5):=(PROBIN(5)*PRUNI)/100.;
        PROBIN(6):=(PROBIN(6)*PRUNI)/100.;

        PRINT(' TEMPS DE POSITIONNEMENT DES UNITES DISQUES ');
        POSIT(1):=GET(REAL);
        PRINT('          ps:1 : ',POSIT(1));
        POSIT(2):=GET(REAL);
        PRINT('          ps:2 : ',POSIT(2));
        POSIT(3):=GET(REAL);
        PRINT('          ps:3 : ',POSIT(3));
        POSIT(4):=GET(REAL);
        PRINT('          us : ',POSIT(4));
        POSIT(5):=GET(REAL);
        PRINT('          montable : ',POSIT(5));
```

```
        PRINT(' TEMPS DE TRANSFERT DES CANAUX ');
        TRANSF(1):=GET(REAL);
        PRINT('          canal0 : ',TRANSF(1));
        TRANSF(2):=GET(REAL);
        PRINT('          canal1 : ',TRANSF(2));
        TRANSF(3):=GET(REAL);
        PRINT('          canal2 : ',TRANSF(3));
        TRANSF(4):=GET(REAL);
        PRINT('          canal3 : ',TRANSF(4));

        PRINT(' NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ ');
        CHANGT(1):=GET(INTEGER);
        PRINT('          ps:1 : ',CHANGT(1));
        IF CHANGT(1) > 1
               THEN CHANGT(1):=CHANGT(1)-1;
        CHANGT(2):=GET(INTEGER);
        PRINT('          ps:2 : ',CHANGT(2));
        IF CHANGT(2) > 1
               THEN CHANGT(2):=CHANGT(2)-1;
        CHANGT(3):=GET(INTEGER);
        PRINT('          ps:3 : ',CHANGT(3));
        IF CHANGT(3) > 1
               THEN CHANGT(3):=CHANGT(3)-1;
        CHANGT(4):=GET(INTEGER);
        PRINT('          us : ',CHANGT(4));
        IF CHANGT(4) > 1
               THEN CHANGT(4):=CHANGT(4)-1;
        CHANGT(5):=GET(INTEGER);
        PRINT('          montable : ',CHANGT(5));
        IF CHANGT(5) > 1
               THEN CHANGT(5):=CHANGT(5)-1;
        PRINT(' ');
        PRINT(' ');
        PRINT(' ');
        PRINT(' ');
        PRINT('*****************************************************************');
        PRINT('*                                                               *');
        PRINT('*        VOICI LES RESULTATS DE LA SIMULATION                   *');
        PRINT('*                                                               *');
        PRINT('*****************************************************************');
        PRINT(' ');
        PRINT(' ');
        SIMUL;
  END;

27          % nbre de proc dans le modele
27          % nbre de proc dans le balance set
2156.5074   % temps moyen de service aux terminaux
10.932015   % temps moyen de service au cpu
26.275   0.679   0.086  72.96              %prob de trans a la sortie du cpu
30.47  24.54  26.65   9.61   0.34   8.39   %pourc pour chaque unite
15.       15.       16.       20.       15.   %positionnement des unites disque
16.       16.       18.       31.             %transfert des canaux
3        3         3         3        14     %iorb transf dePWQ vers TWQ


/END/
```

A N N E X E H

LES RESULTATS DU MODELE AVEC LES PROBABILITES MARGINALES

(prouvant la stationnarité)

RESULTATS AVEC PROBABILITES MARGINALES POUR UNE DUREE DE SIMULATION DE 1.000.000 UNITES

(Les paramètres d'entrée correspondant à la 7ème mesure).

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982


   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          27.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          27.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     2156.5874
 TEMPS MOYEN DE SERVICE DU CPU :     10.932015
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu---->terminaux : 26.275
          cpu---->cpu :  0.679
          cpu---->entree dans balset :  0.086
          cpu---->unites : 72.960
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 30.47
          ps:2 : 24.54
          ps:3 : 26.65
          us :  9.61
          montable :  0.34
          bandes :  8.39
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     15.00
          ps:2 :     15.00
          ps:3 :     16.00
          us :     20.00
          montable :     15.00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     16.00
          canal1 :     16.00
          canal2 :     18.00
          canal3 :     31.00
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
          ps:1 :          3.
          ps:2 :          3.
          ps:3 :          3.
          us :          3.
          montable :          14.
```

```
**********************************************************
*                                                        *
*        VOICI LES RESULTATS DE LA SIMULATION            *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =    100 , CONF. LEVEL = 0.95
*****************************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************************
*           *          *          *          *          *          *
* TTY       * 2167.    * .9999    * 24.92    * 2167.    *     11478*
*   +/-     * 38.85    * .2284E-03* .8725E-01* 38.85     *          *
*           *          *          *          *          *          *
* WAITMEM   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     11543*
*           *          *          *          *          *          *
* LIBTTY    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     11502*
*           *          *          *          *          *          *
* LIBFIL    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        38*
*           *          *          *          *          *          *
* CPU       * 11.02    * .4773    * .8741    * 20.18    *     43306*
*   +/-     * .9711E-01* .1142E-01* .5798E-01* 1.063     *          *
*           *          *          *          *          *          *
* PS1       * 30.82    * .2915    * .3422    * 36.18    *      9459*
*   +/-     * .4350    * .8172E-02* .1203E-01* .7233     *          *
*           *          *          *          *          *          *
* PS2       * 29.87    * .2313    * .2625    * 33.90    *      7744*
*   +/-     * .5269    * .7351E-02* .9722E-02* .6437     *          *
*           *          *          *          *          *          *
* PS3       * 31.69    * .2677    * .3175    * 37.59    *      8447*
*   +/-     * .4918    * .9575E-02* .1443E-01* .8696     *          *
*           *          *          *          *          *          *
* US        * 39.10    * .1198    * .1266    * 41.35    *      3063*
*   +/-     * 1.050    * .5076E-02* .5817E-02* 1.155     *          *
*           *          *          *          *          *          *
* MONT      * 32.27    * .3581E-02* .3595E-02* 32.39     *       111*
*   +/-     * 4.507    * .8254E-03* .8328E-03* 4.529     *          *
*           *          *          *          *          *          *
* CANAL0    * 16.18    * .2026    * .2472    * 19.74    *     12522*
*   +/-     * .2736    * .5761E-02* .8559E-02* .4240     *          *
*           *          *          *          *          *          *
* CANAL1    * 16.39    * .1287    * .1454    * 18.51    *      7855*
*   +/-     * .3464    * .4395E-02* .5883E-02* .4434     *          *
*           *          *          *          *          *          *
* CANAL2    * 18.05    * .1525    * .1774    * 20.99    *      8447*
*   +/-     * .3897    * .6064E-02* .8321E-02* .5645     *          *
*           *          *          *          *          *          *
* CANAL3    * 30.65    * .8071E-01* .8759E-01* 33.27     *      2633*
*   +/-     * 1.229    * .5036E-02* .6108E-02* 1.512     *          *
*           *          *          *          *          *          *
*****************************************************************************
```

```
*** MARGINAL PROBABILITIES : STATION TTY      ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         2.917

                   VARIANCE OF RESPONSE TIME   =   4641240.125

*** MARGINAL PROBABILITIES : STATION SEMMEM   ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000

*** MARGINAL PROBABILITIES : STATION WAITMEM  ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000

*** MARGINAL PROBABILITIES : STATION LIBTTY   ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000
```

```
*** MARGINAL PROBABILITIES : STATION LIBFIL    ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000


*** MARGINAL PROBABILITIES : STATION CPU       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.523 0.259 0.121 0.057 0.025 0.009

                   VARIANCE OF CUSTOMER NUMBER =         1.722

                   VARIANCE OF RESPONSE TIME   =       465.443


*** MARGINAL PROBABILITIES : STATION PS1       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.708 0.248 0.038 0.006 0.001 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.342

                   VARIANCE OF RESPONSE TIME   =       861.790


*** MARGINAL PROBABILITIES : STATION PS2       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.769 0.203 0.025 0.003 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.264

                   VARIANCE OF RESPONSE TIME   =       733.147
```

```
*** MARGINAL PROBABILITIES : STATION PS3       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.732 0.227 0.034 0.006 0.001 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.337

                   VARIANCE OF RESPONSE TIME   =       983.313

*** MARGINAL PROBABILITIES : STATION US        ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.880 0.113 0.006 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.125

                   VARIANCE OF RESPONSE TIME   =       868.166

*** MARGINAL PROBABILITIES : STATION MONT      ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.996 0.004 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.004

                   VARIANCE OF RESPONSE TIME   =       586.179

*** MARGINAL PROBABILITIES : STATION CANALO    ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.797 0.164 0.033 0.005 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.288

                   VARIANCE OF RESPONSE TIME   =       379.190
```

```
*** MARGINAL PROBABILITIES : STATION CANAL1    ***

 GLOBAL          : CUSTNB=      0      1      2      3      4      5
                   PROB=    0.871  0.114  0.013  0.002  0.000  0.000

                   VARIANCE OF CUSTOMER NUMBER =          0.161

                   VARIANCE OF RESPONSE TIME   =        350.350


*** MARGINAL PROBABILITIES : STATION CANAL2    ***

 GLOBAL          : CUSTNB=      0      1      2      3      4      5
                   PROB=    0.848  0.131  0.018  0.003  0.000  0.000

                   VARIANCE OF CUSTOMER NUMBER =          0.202

                   VARIANCE OF RESPONSE TIME   =        445.369


*** MARGINAL PROBABILITIES : STATION CANAL3    ***

 GLOBAL          : CUSTNB=      0      1      2      3      4      5
                   PROB=    0.919  0.074  0.006  0.000  0.000  0.000

                   VARIANCE OF CUSTOMER NUMBER =          0.094

                   VARIANCE OF RESPONSE TIME   =       1135.453


... END OF SIMULATION : TIME = **********

              MEMORY USED:        7607 WORDS OF 4 BYTES
                (  7.61  % OF TOTAL MEMORY)
```

RESULTATS AVEC PROBABILITES MARGINALES POUR UNE DUREE DE SIMULATION DE 1.500.000 UNITES

(Les paramètres d'entrée correspondant à la 7ème mesure).

```
 DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982

    1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDF         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          27.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          27.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     2156.5874
 TEMPS MOYEN DE SERVICE DU CPU :     10.932015
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu--->terminaux : 26.275
          cpu--->cpu :  0.679
          cpu--->entree dans balset :  0.086
          cpu--->unites : 72.960
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 30.47
          ps:2 : 24.54
          ps:3 : 26.65
          us :  9.61
          montable :  0.34
          bandes :  8.39
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     15.00
          ps:2 :     15.00
          ps:3 :     16.00
          us :     20.00
          montable :     15.00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     16.00
          canal1 :     16.00
          canal2 :     18.00
          canal3 :     31.00
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
          ps:1 :          3.
          ps:2 :          3.
          ps:3 :          3.
          us :          3.
          montable :          14.
```

```
**********************************************************
*                                                        *
*       VOICI LES RESULTATS DE LA SIMULATION             *
*                                                        *
**********************************************************
```

*** SIMULATION ***
... TIME = 1500000.00 , NB SAMPLES = 100 , CONF. LEVEL = 0.95

NAME	SERVICE	BUSY PCT	CUST NB	RESPONSE	SERV NB
TTY	2198.	.9999	24.97	2198.	17022
+/-	30.52	.1515E-03	.6667E-01	30.52	
WAITMEM	.0000E+00	.0000E+00	.0000E+00	.0000E+00	17097
LIBTTY	.0000E+00	.0000E+00	.0000E+00	.0000E+00	17046
LIBFIL	.0000E+00	.0000E+00	.0000E+00	.0000E+00	48
CPU	11.05	.4696	.8473	19.94	63754
+/-	.9310E-01	.9457E-02	.4233E-01	.7730	
PS1	30.72	.2858	.3353	36.04	13954
+/-	.3752	.7180E-02	.1038E-01	.6175	
PS2	29.82	.2264	.2562	33.74	11387
+/-	.4378	.6241E-02	.8076E-02	.5524	
PS3	32.01	.2642	.3114	37.73	12380
+/-	.3898	.7606E-02	.1102E-01	.6358	
US	39.12	.1161	.1228	41.36	4452
+/-	.7572	.4335E-02	.5037E-02	.8698	
MONT	32.45	.3721E-02	.3730E-02	32.53	172
+/-	3.635	.7737E-03	.7817E-03	3.696	
CANAL0	16.07	.1972	.2405	19.60	18406
+/-	.2468	.5255E-02	.7922E-02	.3972	
CANAL1	16.19	.1248	.1406	18.24	11559
+/-	.2882	.3825E-02	.4964E-02	.3796	
CANAL2	18.26	.1507	.1747	21.17	12380
+/-	.3192	.4779E-02	.6309E-02	.4311	
CANAL3	30.60	.7902E-01	.8554E-01	33.13	3873
+/-	1.061	.4031E-02	.4829E-02	1.274	

```
*** MARGINAL PROBABILITIES : STATION TTY      ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         2.748

                   VARIANCE OF RESPONSE TIME   =   4741758.375


*** MARGINAL PROBABILITIES : STATION SEMMEM   ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000


*** MARGINAL PROBABILITIES : STATION WAITMEM  ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000


*** MARGINAL PROBABILITIES : STATION LIBTTY   ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000
```

```
*** MARGINAL PROBABILITIES : STATION LIBFIL    ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   1.000 0.000 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.000

                   VARIANCE OF RESPONSE TIME   =         0.000


*** MARGINAL PROBABILITIES : STATION CPU       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.530 0.257 0.119 0.055 0.024 0.008

                   VARIANCE OF CUSTOMER NUMBER =         1.577

                   VARIANCE OF RESPONSE TIME   =       433.939


*** MARGINAL PROBABILITIES : STATION PS1       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.714 0.243 0.037 0.005 0.001 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.337

                   VARIANCE OF RESPONSE TIME   =       859.139


*** MARGINAL PROBABILITIES : STATION PS2       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.774 0.200 0.023 0.003 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.258

                   VARIANCE OF RESPONSE TIME   =       712.398
```

```
*** MARGINAL PROBABILITIES : STATION FS3      ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.736 0.225 0.033 0.005 0.001 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.327

                   VARIANCE OF RESPONSE TIME   =       958.297


*** MARGINAL PROBABILITIES : STATION US       ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.884 0.110 0.006 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.121

                   VARIANCE OF RESPONSE TIME   =       884.441


*** MARGINAL PROBABILITIES : STATION MONT     ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.996 0.004 0.000 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.004

                   VARIANCE OF RESPONSE TIME   =       563.400


*** MARGINAL PROBABILITIES : STATION CANALO   ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.803 0.160 0.032 0.005 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.283

                   VARIANCE OF RESPONSE TIME   =       374.011
```

```
*** MARGINAL PROBABILITIES : STATION CANAL1    ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.875 0.111 0.013 0.002 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.156

                   VARIANCE OF RESPONSE TIME   =       334.871


*** MARGINAL PROBABILITIES : STATION CANAL2    ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.849 0.130 0.018 0.003 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.198

                   VARIANCE OF RESPONSE TIME   =       445.879


*** MARGINAL PROBABILITIES : STATION CANAL3    ***

 GLOBAL          : CUSTNB=     0     1     2     3     4     5
                   PROB=   0.921 0.073 0.006 0.000 0.000 0.000

                   VARIANCE OF CUSTOMER NUMBER =         0.092

                   VARIANCE OF RESPONSE TIME   =      1103.613

... END OF SIMULATION : TIME = **********

              MEMORY USED:        7607 WORDS OF 4 BYTES
               (  7.61  % OF TOTAL MEMORY)
```

A N N E X E I

LES RESULTATS DU MODELE

PREMIERE SIMULATION

(correspondant à la première mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982

   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          28.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          28.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     1966.5555
 TEMPS MOYEN DE SERVICE DU CPU :      8.438268
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu----->terminaux : 23.598
          cpu----->cpu :  0.488
          cpu----->entree dans balset :  0.106
          cpu----->unites : 75.808
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 26.74
          ps:2 : 27.55
          ps:3 : 24.15
          us :  3.61
          montable :  0.06
          bandes : 17.90
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     18.00
          ps:2 :     18.00
          ps:3 :     19.00
          us :    18.00
          montable :     10.00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     17.00
          canal1 :     17.00
          canal2 :     18.00
          canal3 :     54.00
 NOMBRE D'IORB's TRANSFERES DE LA FWQ VERS LA TWQ
          ps:1 :          3.
          ps:2 :          3.
          ps:3 :          3.
          us :         3.
          montable :          3.
```

```
**********************************************************
*                                                        *
*       VOICI LES RESULTATS DE LA SIMULATION             *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =    100 , CONF. LEVEL = 0.95
*****************************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************************
*           *          *          *          *          *          *
* TTY       * 1994.    * .9999    * 25.20    * 1994.    *     12619*
*   +/-     * 34.78    * .1114E-03* .1085    * 34.78    *          *
*           *          *          *          *          *          *
* WAITMEM   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     12695*
*           *          *          *          *          *          *
* LIBTTY    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     12638*
*           *          *          *          *          *          *
* LIBFIL    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        48*
*           *          *          *          *          *          *
* CPU       * 8.533    * .4494    * .7780    * 14.77    *     52669*
*   +/-     * .6871E-01* .9296E-02* .3916E-01* .5361    *          *
*           *          *          *          *          *          *
* PS1       * 32.67    * .3453    * .4257    * 40.28    *     10569*
*   +/-     * .4421    * .1165E-01* .1919E-01* .8841    *          *
*           *          *          *          *          *          *
* PS2       * 31.73    * .3502    * .4341    * 39.33    *     11038*
*   +/-     * .4715    * .1047E-01* .1673E-01* .8357    *          *
*           *          *          *          *          *          *
* PS3       * 33.83    * .3244    * .3916    * 40.83    *      9590*
*   +/-     * .5531    * .8992E-02* .1360E-01* .7797    *          *
*           *          *          *          *          *          *
* US        * 41.63    * .5779E-01* .5932E-01* 42.74    *      1388*
*   +/-     * 1.597    * .3963E-02* .4144E-02* 1.663    *          *
*           *          *          *          *          *          *
* MONT      * 39.38    * .6694E-03* .6694E-03* 39.38    *        17*
*   +/-     * 9.650    * .3552E-03* .3552E-03* 9.650    *          *
*           *          *          *          *          *          *
* CANAL0    * 17.09    * .2043    * .2550    * 21.33    *     11957*
*   +/-     * .3035    * .7462E-02* .1204E-01* .5697    *          *
*           *          *          *          *          *          *
* CANAL1    * 16.97    * .1876    * .2274    * 20.57    *     11055*
*   +/-     * .3637    * .7152E-02* .1001E-01* .5368    *          *
*           *          *          *          *          *          *
* CANAL2    * 17.88    * .1715    * .2038    * 21.25    *      9590*
*   +/-     * .3687    * .5659E-02* .7509E-02* .4703    *          *
*           *          *          *          *          *          *
* CANAL3    * 54.18    * .3858    * .5997    * 84.22    *      7121*
*   +/-     * 1.292    * .1330E-01* .4077E-01* 4.474    *          *
*           *          *          *          *          *          *
*****************************************************************************
... END OF SIMULATION : TIME = **********

                MEMORY USED:        7523 WORDS OF 4 BYTES
                 (  7.52  % OF TOTAL MEMORY)
```

DEUXIEME SIMULATION

(correspondant à la deuxième mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982


   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          30.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          30.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     1826.5049
 TEMPS MOYEN DE SERVICE DU CPU :      9.337532
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu------>terminaux : 24.456
          cpu---->cpu :  0.553
          cpu---->entree dans balset :  0.075
          cpu---->unites : 74.916
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 32.27
          ps:2 : 25.57
          ps:3 : 25.41
          us : 15.89
          montable :  0.01
          bandes :  0.85
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     16.00
          ps:2 :     15.00
          ps:3 :     17.00
          us :     21.00
          montable :     11.00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     16.00
          canal1 :     17.00
          canal2 :     18.00
          canal3 :     183.0
 NOMBRE D'IORB's TRANSFERES DE LA FWQ VERS LA TWQ
          ps:1 :          2.
          ps:2 :          2.
          ps:3 :          3.
          us :          5.
          montable :          3.
```

```
**********************************************************
*                                                        *
*       VOICI LES RESULTATS DE LA SIMULATION             *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =     100 , CONF. LEVEL = 0.95
**************************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
**************************************************************************
*           *          *          *          *          *          *
* TTY       * 1849.    * 1.000    * 26.78    * 1849.    *     14449*
*   +/-     * 33.02    * .4457E-04* .1136    * 33.02    *          *
*           *          *          *          *          *          *
* WAITMEM   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     14537*
*           *          *          *          *          *          *
* LIBTTY    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     14478*
*           *          *          *          *          *          *
* LIBFIL    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        58*
*           *          *          *          *          *          *
* CPU       * 9.475    * .5548    * 1.155    * 19.72    *     58547*
*   +/-     * .7804E-01* .1154E-01* .5835E-01* .7530    *          *
*           *          *          *          *          *          *
* PS1       * 33.11    * .4638    * .6594    * 47.08    *     14007*
*   +/-     * .3751    * .1192E-01* .2752E-01* 1.151    *          *
*           *          *          *          *          *          *
* PS2       * 29.96    * .3371    * .4218    * 37.49    *     11251*
*   +/-     * .3936    * .9162E-02* .1638E-01* .8231    *          *
*           *          *          *          *          *          *
* PS3       * 31.97    * .3535    * .4375    * 39.56    *     11059*
*   +/-     * .3885    * .9528E-02* .1503E-01* .6230    *          *
*           *          *          *          *          *          *
* US        * 40.22    * .2814    * .3277    * 46.83    *      6997*
*   +/-     * .6823    * .1032E-01* .1413E-01* .9586    *          *
*           *          *          *          *          *          *
* MONT      * 27.80    * .8340E-04* .8340E-04* 27.80    *         3*
*   +/-     * 12.01    * .1001E-03* .1001E-03* 12.01    *          *
*           *          *          *          *          *          *
* CANAL0    * 16.15    * .3392    * .4755    * 22.64    *     21004*
*   +/-     * .2456    * .9085E-02* .1746E-01* .4931    *          *
*           *          *          *          *          *          *
* CANAL1    * 17.08    * .1922    * .2214    * 19.67    *     11254*
*   +/-     * .3019    * .5971E-02* .7815E-02* .4070    *          *
*           *          *          *          *          *          *
* CANAL2    * 18.06    * .1997    * .2402    * 21.72    *     11059*
*   +/-     * .3001    * .6394E-02* .9326E-02* .4584    *          *
*           *          *          *          *          *          *
* CANAL3    * 174.4    * .6521E-01* .6902E-01* 184.6    *       374*
*   +/-     * 16.00    * .8836E-02* .9671E-02* 17.63    *          *
*           *          *          *          *          *          *
**************************************************************************
... END OF SIMULATION : TIME = **********

                MEMORY USED:        7596 WORDS OF 4 BYTES
                 (  7.60  % OF TOTAL MEMORY)
```

TROISIEME SIMULATION

(correspondant à la troisième mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982


   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDF         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          24.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          24.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     2055.9495
 TEMPS MOYEN DE SERVICE DU CPU :     11.475786
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu---->terminaux : 24.623
          cpu---->cpu :  0.722
          cpu--->entree dans balset :  0.122
          cpu---->unites : 74.533
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 32.33
          ps:2 : 30.44
          ps:3 : 29.19
          us :  7.04
          montable :  0.00
          bandes :  1.00
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     16.00
          ps:2 :     18.00
          ps:3 :     17.00
          us :     22.00
          montable :    0.0000E+00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     16.00
          canal1 :     17.00
          canal2 :     18.00
          canal3 :     199.0
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
          ps:1 :          3.
          ps:2 :          3.
          ps:3 :          3.
          us :          2.
          montable :          0.
```

```
**********************************************************
*                                                        *
*        VOICI LES RESULTATS DE LA SIMULATION            *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =    100 , CONF. LEVEL = 0.95
**************************************************************************
*   NAME   *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
**************************************************************************
*          *          *          *          *          *          *
* TTY      * 2096.    * 1.000    * 21.81    * 2096.    *     10383*
*   +/-    * 40.81    * .4457E-04* .7219E-01* 40.81    *          *
*          *          *          *          *          *          *
* WAITMEM  * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     10465*
*          *          *          *          *          *          *
* LIBTTY   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     10407*
*          *          *          *          *          *          *
* LIBFIL   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        58*
*          *          *          *          *          *          *
* CPU      * 11.63    * .4848    * .8687    * 20.84    *     41684*
*   +/-    * .1066    * .1008E-01* .3727E-01* .6161    *          *
*          *          *          *          *          *          *
* PS1      * 31.36    * .3100    * .3703    * 37.47    *      9885*
*   +/-    * .4950    * .9525E-02* .1410E-01* .8027    *          *
*          *          *          *          *          *          *
* PS2      * 32.22    * .3043    * .3621    * 38.34    *      9445*
*   +/-    * .4576    * .8654E-02* .1251E-01* .6756    *          *
*          *          *          *          *          *          *
* PS3      * 32.55    * .2932    * .3455    * 38.35    *      9009*
*   +/-    * .4460    * .7762E-02* .1176E-01* .6648    *          *
*          *          *          *          *          *          *
* US       * 41.78    * .9416E-01* .9857E-01* 43.73    *      2254*
*   +/-    * 1.315    * .5755E-02* .6306E-02* 1.490    *          *
*          *          *          *          *          *          *
* CANAL0   * 16.19    * .1965    * .2407    * 19.83    *     12139*
*   +/-    * .2703    * .6018E-02* .9039E-02* .4407    *          *
*          *          *          *          *          *          *
* CANAL1   * 17.03    * .1608    * .1891    * 20.02    *      9445*
*   +/-    * .3248    * .5527E-02* .7559E-02* .4848    *          *
*          *          *          *          *          *          *
* CANAL2   * 18.06    * .1627    * .1890    * 20.98    *      9009*
*   +/-    * .3570    * .5368E-02* .7085E-02* .4603    *          *
*          *          *          *          *          *          *
* CANAL3   * 179.0    * .5889E-01* .6222E-01* 189.1    *       329*
*   +/-    * 20.37    * .8580E-02* .9683E-02* 23.13    *          *
*          *          *          *          *          *          *
**************************************************************************
... END OF SIMULATION : TIME = **********

                MEMORY USED:        7394 WORDS OF 4 BYTES
                 (  7.39  % OF TOTAL MEMORY)
```

QUATRIEME SIMULATION

(correspondant à la quatrième mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982

   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          32.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          32.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     2024.6933
 TEMPS MOYEN DE SERVICE DU CPU :      9.423065
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu--->terminaux : 31.605
          cpu--->cpu :  0.560
          cpu--->entree dans balset :  0.060
          cpu--->unites : 67.775
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 35.38
          ps:2 : 22.85
          ps:3 : 29.34
          us :  7.33
          montable :  0.06
          bandes :  5.04
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     15.00
          ps:2 :     15.00
          ps:3 :     17.00
          us :     20.00
          montable :     19.00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     16.00
          canal1 :     18.00
          canal2 :     16.00
          canal3 :     50.00
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
          ps:1 :          2.
          ps:2 :          2.
          ps:3 :          2.
          us :          4.
          montable :          9.
```

```
**********************************************************
*                                                        *
*        VOICI LES RESULTATS DE LA SIMULATION            *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =     100 , CONF. LEVEL = 0.95
*****************************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************************
*           *          *          *          *          *          *
* TTY       * 2070.    * 1.000    * 30.04    * 2070.    *     14486*
*   +/-     * 34.82    * .3152E-04* .7254E-01* 34.82    *          *
*           *          *          *          *          *          *
* WAITMEM   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     14554*
*           *          *          *          *          *          *
* LIBTTY    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     14515*
*           *          *          *          *          *          *
* LIBFIL    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        36*
*           *          *          *          *          *          *
* CPU       * 9.475    * .4295    * .7265    * 16.03    *     45326*
*   +/-     * .8951E-01* .9446E-02* .3561E-01* .5667    *          *
*           *          *          *          *          *          *
* PS1       * 30.23    * .3281    * .4064    * 37.44    *     10854*
*   +/-     * .4335    * .9691E-02* .1585E-01* .7894    *          *
*           *          *          *          *          *          *
* PS2       * 31.74    * .2210    * .2538    * 36.46    *      6962*
*   +/-     * .4892    * .7766E-02* .1123E-01* .7586    *          *
*           *          *          *          *          *          *
* PS3       * 30.74    * .2759    * .3310    * 36.88    *      8976*
*   +/-     * .4402    * .9208E-02* .1461E-01* 1.007    *          *
*           *          *          *          *          *          *
* US        * 40.83    * .8912E-01* .9297E-01* 42.59    *      2183*
*   +/-     * 1.359    * .5139E-02* .5616E-02* 1.415    *          *
*           *          *          *          *          *          *
* MONT      * 45.96    * .8732E-03* .8732E-03* 45.96    *        19*
*   +/-     * 11.39    * .4604E-03* .4604E-03* 11.39    *          *
*           *          *          *          *          *          *
* CANAL 0   * 16.04    * .2092    * .2512    * 19.26    *     13037*
*   +/-     * .2851    * .6844E-02* .9802E-02* .4195    *          *
*           *          *          *          *          *          *
* CANAL1    * 18.04    * .1259    * .1390    * 19.92    *      6981*
*   +/-     * .3885    * .4933E-02* .6014E-02* .4622    *          *
*           *          *          *          *          *          *
* CANAL 2   * 16.17    * .1451    * .1638    * 18.25    *      8976*
*   +/-     * .3522    * .5617E-02* .6902E-02* .4383    *          *
*           *          *          *          *          *          *
* CANAL 3   * 50.62    * .7776E-01* .8336E-01* 54.27    *      1536*
*   +/-     * 2.785    * .6037E-02* .7136E-02* 3.258    *          *
*           *          *          *          *          *          *
*****************************************************************************
... END OF SIMULATION : TIME = **********

                MEMORY USED:        7665 WORDS OF 4 BYTES
                 (  7.67  % OF TOTAL MEMORY)
```

CINQUIEME SIMULATION

(correspondant à la cinquième mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982

   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          44.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          44.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     3547.1248
 TEMPS MOYEN DE SERVICE DU CPU :     11.467859
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu----->terminaux : 31.542
          cpu----->cpu :  0.614
          cpu----->entree dans balset :  0.078
          cpu----->unites : 67.766
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 35.42
          ps:2 : 29.58
          ps:3 : 30.05
          us :  2.91
          montable :  0.03
          bandes :  2.01
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     16.00
          ps:2 :     14.00
          ps:3 :     16.00
          us :     23.00
          montable :     5.000
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     18.00
          canal1 :     23.00
          canal2 :     19.00
          canal3 :     183.0
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
          ps:1 :          3.
          ps:2 :          3.
          ps:3 :          3.
          us :          2.
          montable :          3.
```

```
**********************************************************
*                                                        *
*        VOICI LES RESULTATS DE LA SIMULATION            *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =     100 , CONF. LEVEL = 0.95
*****************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*           *          *          *          *          *          *
* TTY       * 3619.    * .9999    * 42.12    * 3619.    *     11601*
*   +/-     * 70.07    * .1045E-03* .9583E-01* 70.07     *          *
*           *          *          *          *          *          *
* WAITMEM   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     11680*
*           *          *          *          *          *          *
* LIBTTY    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     11641*
*           *          *          *          *          *          *
* LIBFIL    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        35*
*           *          *          *          *          *          *
* CPU       * 11.57    * .4259    * .7356    * 19.98    *     36811*
*   +/-     * .1325    * .8913E-02* .6406E-01* 1.533     *          *
*           *          *          *          *          *          *
* PS1       * 32.59    * .2891    * .3458    * 38.97    *      8872*
*   +/-     * .5333    * .8614E-02* .1377E-01* .9866     *          *
*           *          *          *          *          *          *
* PS2       * 35.47    * .2589    * .2997    * 41.06    *      7298*
*   +/-     * .6641    * .7641E-02* .1073E-01* 1.006     *          *
*           *          *          *          *          *          *
* PS3       * 33.23    * .2473    * .2844    * 38.21    *      7443*
*   +/-     * .5394    * .8616E-02* .1218E-01* .8246     *          *
*           *          *          *          *          *          *
* US        * 44.89    * .3416E-01* .3497E-01* 45.95     *       761*
*   +/-     * 2.206    * .2941E-02* .3121E-02* 2.364     *          *
*           *          *          *          *          *          *
* MONT      * 35.61    * .2849E-03* .2849E-03* 35.61     *         8*
*   +/-     * 20.16    * .2494E-03* .2494E-03* 20.16     *          *
*           *          *          *          *          *          *
* CANAL0    * 18.27    * .1760    * .2087    * 21.67    *      9633*
*   +/-     * .3932    * .6347E-02* .9369E-02* .5927     *          *
*           *          *          *          *          *          *
* CANAL1    * 23.27    * .1700    * .1911    * 26.16    *      7306*
*   +/-     * .5915    * .6154E-02* .7992E-02* .7720     *          *
*           *          *          *          *          *          *
* CANAL2    * 18.97    * .1412    * .1612    * 21.66    *      7443*
*   +/-     * .4287    * .5781E-02* .7580E-02* .5715     *          *
*           *          *          *          *          *          *
* CANAL3    * 176.8    * .9423E-01* .1064    * 199.6    *       533*
*   +/-     * 14.20    * .1180E-01* .1586E-01* 21.19     *          *
*           *          *          *          *          *          *
*****************************************************************
... END OF SIMULATION : TIME = **********

          MEMORY USED:        8038 WORDS OF 4 BYTES
           (  8.04  % OF TOTAL MEMORY)
```

SIXIEME SIMULATION

(correspondant à la sixième mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982

   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          30.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          30.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     2070.8829
 TEMPS MOYEN DE SERVICE DU CPU :      8.542856
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
         cpu----->terminaux : 23.151
         cpu----->cpu :  0.484
         cpu----->entree dans balset :  0.101
         cpu----->unites : 76.264
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
         ps:1 : 30.59
         ps:2 : 27.37
         ps:3 : 28.85
         us :  9.77
         montable :  0.01
         bandes :  3.42
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
         ps:1 :     16.00
         ps:2 :     18.00
         ps:3 :     18.00
         us :     19.00
         montable :     9.000
 TEMPS DE TRANSFERT DES CANAUX
         canal0 :     15.00
         canal1 :     16.00
         canal2 :     16.00
         canal3 :     89.00
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
         ps:1 :          3.
         ps:2 :          3.
         ps:3 :          3.
         us :          4.
         montable :          3.
```

```
**********************************************************
*                                                        *
*        VOICI LES RESULTATS DE LA SIMULATION            *
*                                                        *
**********************************************************


*** SIMULATION ***
... TIME =  1000000.00 , NB SAMPLES =    100 , CONF. LEVEL = 0.95
*****************************************************************
*   NAME    *  SERVICE * BUSY PCT *  CUST NB * RESPONSE *  SERV NB *
*****************************************************************
*           *          *          *          *          *          *
* TTY       * 2098.    * 1.000    * 27.22    * 2098.    *     12948*
*   +/-     * 42.06    * .3152E-04* .1084    * 42.06    *          *
*           *          *          *          *          *          *
* WAITMEM   * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     13027*
*           *          *          *          *          *          *
* LIBTTY    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*     12976*
*           *          *          *          *          *          *
* LIBFIL    * .0000E+00* .0000E+00* .0000E+00* .0000E+00*        49*
*           *          *          *          *          *          *
* CPU       * 8.608    * .4778    * .8821    * 15.89    *     55509*
*   +/-     * .8529E-01* .1097E-01* .4416E-01* .5469    *          *
*           *          *          *          *          *          *
* PS1       * 30.35    * .3880    * .5006    * 39.15    *     12784*
*   +/-     * .3965    * .1163E-01* .2160E-01* .8593    *          *
*           *          *          *          *          *          *
* PS2       * 30.75    * .3556    * .4441    * 38.40    *     11565*
*   +/-     * .3874    * .1023E-01* .1660E-01* .7131    *          *
*           *          *          *          *          *          *
* PS3       * 30.64    * .3775    * .4797    * 38.94    *     12319*
*   +/-     * .4249    * .1123E-01* .2056E-01* .9086    *          *
*           *          *          *          *          *          *
* US        * 39.64    * .1614    * .1748    * 42.93    *      4072*
*   +/-     * .8758    * .7706E-02* .9333E-02* 1.132    *          *
*           *          *          *          *          *          *
* MONT      * 40.65    * .2033E-03* .2033E-03* 40.65    *         5*
*   +/-     * 11.81    * .1843E-03* .1843E-03* 11.81    *          *
*           *          *          *          *          *          *
* CANAL0    * 15.35    * .2587    * .3501    * 20.77    *     16856*
*   +/-     * .2268    * .8103E-02* .1611E-01* .5461    *          *
*           *          *          *          *          *          *
* CANAL1    * 16.17    * .1871    * .2292    * 19.81    *     11570*
*   +/-     * .2789    * .6795E-02* .9500E-02* .4266    *          *
*           *          *          *          *          *          *
* CANAL2    * 16.25    * .2002    * .2493    * 20.23    *     12319*
*   +/-     * .2903    * .6715E-02* .1067E-01* .5071    *          *
*           *          *          *          *          *          *
* CANAL3    * 86.64    * .1293    * .1494    * 100.2    *      1492*
*   +/-     * 4.505    * .1043E-01* .1510E-01* 7.418    *          *
*           *          *          *          *          *          *
*****************************************************************
... END OF SIMULATION : TIME = **********

                MEMORY USED:        7592 WORDS OF 4 BYTES
                 (  7.59  % OF TOTAL MEMORY)
```

SEPTIEME SIMULATION

(correspondant à la septième mesure)

```
DEC 20/60 *** QNAP2.1 *** (FEBRUARY 83)
(C) COPYRIGHT BY CII HONEYWELL BULL AND INRIA, 1982

   1    /CONTROL / OPTION=NSOURCE;

*****************************************************************
*                                                               *
*    CE PROGRAMME EFFECTUE LA SIMULATION DU DEC-20 FNDP         *
*              AVEC LES PARAMETRES SUIVANTS                     *
*                                                               *
*****************************************************************


 NOMBRE DE PROCESSUS DANS LE MODELE :          27.
 NOMBRE MOYEN DE PROCESSUS DANS LE BALANCE SET :          27.
 TEMPS MOYEN DE SERVICE DES TERMINAUX :     2156.5874
 TEMPS MOYEN DE SERVICE DU CPU :     10.932015
 PROBABILITES DE TRANSITION A LA SORTIE DU CPU
          cpu----->terminaux : 26.275
          cpu---->cpu :  0.679
          cpu---->entree dans balset :  0.086
          cpu----->unites : 72.960
 POUCENTAGE POUR CHAQUE UNITE INDIVIDUELLEMENT
          ps:1 : 30.47
          ps:2 : 24.54
          ps:3 : 26.65
          us :  9.61
          montable :  0.34
          bandes :  8.39
 TEMPS DE POSITIONNEMENT DES UNITES DISQUES
          ps:1 :     15.00
          ps:2 :     15.00
          ps:3 :     16.00
          us :     20.00
          montable :     15.00
 TEMPS DE TRANSFERT DES CANAUX
          canal0 :     16.00
          canal1 :     16.00
          canal2 :     18.00
          canal3 :     31.00
 NOMBRE D'IORB's TRANSFERES DE LA PWQ VERS LA TWQ
          ps:1 :          3.
          ps:2 :          3.
          ps:3 :          3.
          us :          3.
          montable :          14.
```