Entrées/Sorties dans les système d'exploitation

Luc Finet

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR DE TROISIEME CYCLE

"Informatique"

par

Luc FINET

ENTREES - SORTIES

DANS LES SYSTEMES D'EXPLOITATION

Thèse soutenue le 30 Juin 1973 devant la commission d'examen

PRESIDENT : MONSIEUR N. GASTINEL

EXAMINATEURS : MESSIEURS L. BOLLIET

J. BELLINO

S. KRAKOWIAK

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

LISTE DES PROFESSEURS

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOUD Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BENOIT Jean	Radioélectricité
	BERNARD Alain	Mathématiques Pures
	BESSON Jean	Electrochimie
	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BONNIER Etienne	Electrochimie Electrométallurgie
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BRAVARD Yves	Géographie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	CABANAC Jean	Pathologie chirurgicale
	CABANEL Jean	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHATEAU Robert	Thérapeutique
	CHENE Marcel	Chimie papetière
	COEUR André	Pharmacie chimique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique

Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	FAU René	Clinique neuro-psychiatrique
	FELICI Noël	Electrostatique
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GIRAUD Pierre	Géologie
	KLEIN Joseph	Mathématiques Pures
Mme	KOFLER Lucie	Botanique et Physilogie végétale
MM.	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MASSEPORT Jean	Géographie
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAUTHENET René	Electrotechnique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET René	Servomécanismes
	PILLET Emile	Physique industrielle
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	SANTON Lucien	Mécanique
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SILBERT Robert	Mécanique des fluides
	SOUTIF Michel	Physique générale

MM.	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLAND François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
M.	VERAIN André	Physique
Mme	VEYRET Germaine	Géographie
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	BULLEMER Bernhard	Physique
	HANO JUN-ICHI	Mathématiques Pures
	STEPHENS Michaël	Mathématiques appliquées

PROFESSEURS SANS CHAIRE

MM.	BEAUDOING André	Pédiatrie
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BONNETAIN Lucien	Chimie minérale
Mme	BONNIER Jane	Chimie générale
MM.	CARLIER Georges	Biologie végétale
	COHEN Joseph	Electrotechnique
	COUMES André	Radioélectricité
	DEPASSEL Roger	Mécanique des fluides
	DEPORTES Charles	Chimie minérale
	GAUTHIER Yves	Sciences biologiques
	GAVEND Michel	Pharmacologie
	GERMAIN Jean-Pierre	Mécanique
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	JANIN Bernard	Géographie
Mme	KAHANE Josette	Physique
MM.	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	POULOUJADOFF Michel	Electrotechnique
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	DE ROUGEMONT Jacques	Neurochirurgie
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie
	BEGUIN Claude	Chimie organique
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BILLET Jean	Géographie
	BLIMAN Samuel	Electronique (EIE)
	BLOCH Daniel	Electrotechnique
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BOUVARD Maurice	Mécanique des fluides
	BRODEAU François	Mathématiques (IUT B)
	BRUGEL Lucien	Energétique
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CHIBON Pierre	Biologie animale
	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	CONTE René	Physique
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	DURAND Francis	Métallurgie
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GRIFFITHS Michaël	Mathématiques appliquées
	GROULADE Joseph	Biochimie médicale
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine préventive
	IDELMAN Simon	Physiologie animale
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	JOLY Jean-René	Mathématiques Pures
	JOUBERT Jean-Claude	Physique du solide
	JULLIEN Pierre	Mathématiques Pures
	KAHANE André	Physique générale
	KUHN Gérard	Physique
	LACOUME Jean-Louis	Physique
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LANCIA Roland	Physique atomique
	LE JUNTER Noël	Electronique
	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique nucléaire
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LUU DUC Cuong	Chimie organique
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)

MM.	MAYNARD Roger	Physique du solide
	MICHOULIER Jean	Physique (IUT A)
	MICOUD Max	Maladies infectieuses
	MOREAU René	Hydraulique (INP)
	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PERRIN Louis	Pathologie expérimentale
	PFISTER Jean-Claude	Physique du solide
	PHELIP Xavier	Rhumatologie
Mle	RIERY Yvette	Biologie animale
MM.	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RENAUD Maurice	Chimie
	RICHARD Lucien	Botanique
Mme	RINAUDO Marquerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	SHOM Jean-Claude	Chimie générale
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VAN CUTSEM Bernard	Mathématiques appliquées
	VEILLON Gérard	Mathématiques appliquées (INP)
	VIALON Pierre	Géologie
	VOOG Robert	Médecine interne
	VROUSSOS Constantin	Radiologie
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES ASSOCIES

MM.	BOUDOURIS Georges	Radioélectricité
	CHEEKE John	Thermodynamique
	GOLDSCHMIDT Hubert	Mathématiques
	SIDNEY STUARD	Mathématiques Pures
	YACOUD Mahmoud	Médecine légale

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

Mme	BERIEL Hélène	Physilogie
Mme	RENAUDET Jacqueline	Microbiologie

Fait le 30 mai 1972.

Je tiens à remercier :

Monsieur Noël GASTINEL qui a bien voulu me faire l'honneur de présider le jury de cette thèse,

Monsieur Louis BOLLIET, qui a toujours montré la plus grande bienveillance pour mon travail,

Monsieur Sacha KRAKOWIAK qui s'est intéressé à mon travail et a bien voulu accepter de le juger,

Je tiens à remercier plus particulièrement Monsieur Jacques BELLINO qui m'a constamment guidé et conseillé et à qui je dois d'avoir pu mener à bien ce travail,

Je voudrais remercier aussi Monsieur Philippe POTIN qui n'a pas ménagé son temps pour m'aider dans mon travail,

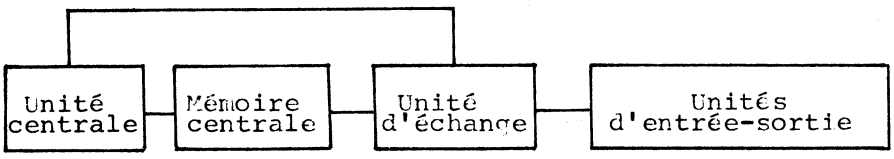
et tous mes collègues du Laboratoire de Calcul et du Centre Scientifique IBM qui m'ont aidé dans la réalisation de ma tâche grâce à leurs remarques ou leur contribution active, en particulier Messieurs HANS, de LAMBERTERIE, LEROUDIER.

Je n'oublie pas non plus tous ceux et celles qui ont contribué à la réalisation matérielle de cet ouvrage.

TABLE DES MATIERES

INTRODUCTION

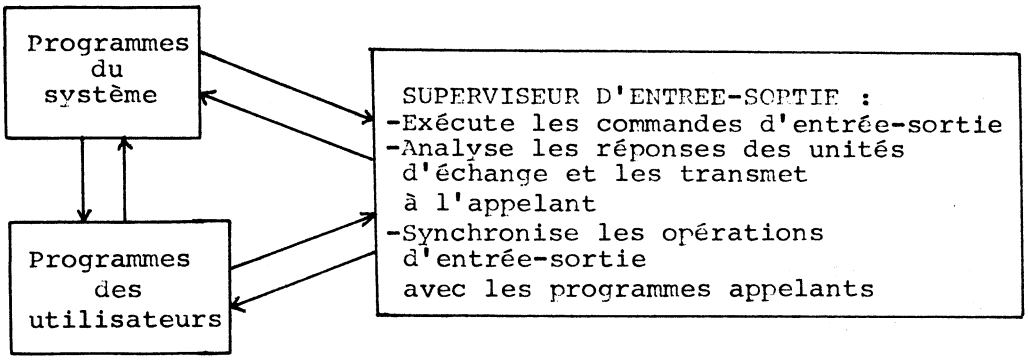
Les opérations d'entrée-sortie dans les ordinateurs recouvrent tout ce qui se rapporte aux transferts d'informations entre la mémoire centrale et les unités périphériques. Elles sont à la base des communications entre l'homme et l'ordinateur : une console, un lecteur de cartes transmettent des informations directement assimilables par l'ordinateur ; dans un système en temps réel, la communication peut se faire entre l'ordinateur et des machines destinées à émettre ou à recevoir des signaux (commande de machines-outils, guidage de satellites). Les unités d'entrée-sortie sont aussi très employées pour la conservation d'informations, pendant une durée variant de quelques secondes à plusieurs semaines : les tambours, les disques, les bandes sont utilisées dans ce but. L'unité centrale de traitement, qui exécute les instructions des programmes qu'on lui soumet, ne peut accéder qu'aux données contenues en mémoire centrale. Dans la plupart des ordinateurs actuels, les unités d'entrée-sortie sont connectées à la mémoire centrale par l'intermédiaire d'unités d'échange. Elles exécutent des commandes élémentaires différentes des instructions de l'unité centrale : par exemple, écrire une ligne sur une imprimante.



Ces commandes réalisent les transferts de données entre la mémoire centrale et les unités d'entrée-sortie. Généralement, les unités

d'échange sont conçues pour pouvoir fonctionner en simultanéité avec l'unité centrale. Il y a alors parallélisme total ou partiel entre les opérations d'entrée-sortie et l'exécution des instructions de l'unité centrale. Toutefois, les unités d'échange ne travaillent que sur ordre de l'unité centrale. Il doit donc y avoir en mémoire centrale des programmes qui gèrent les unités d'échange et les unités. Dans beaucoup de systèmes, ces programmes sont regroupés dans un module unique, le superviseur d'entrée-sortie . Le besoin d'un superviseur d'entrée-sortie est dû au fait que les ressources d'entrée-sortie (unités d'échange, unités) ne sont pas en général allouées à un seul programme, mais utilisées par plusieurs programmes indépendants. De ce point de vue, le superviseur d'entrée-sortie est la partie du système qui contrôle l'allocation des ressources d'entrée-sortie. Le superviseur d'entrée-sortie reçoit de divers points du système des demandes d'entrée-sortie : à chacune de ces demandes correspond une commande élémentaire ou une suite de commandes élémentaires, à exécuter sur une unité donnée. Pour exécuter ces commandes, le superviseur d'entrée-sortie active l'unité d'échange, et analyse les réponses de l'unité d'échange et de l'unité aux commandes. Il tente de corriger les erreurs éventuelles. Il établit des protections d'accès quand elles peuvent l'être au niveau des commandes : par exemple empêcher toute opération d'écriture sur un disque. Une autre fonction qui est la sienne est d'établir des liens entre la demande d'entrée-sortie et le programme qui l'a émise. Il synchronise la demande et le programme, et peut fournir à ce dernier des informations sur la façon dont s'est déroulée l'entrée-sortie. Un superviseur d'entrée-sortie peut laisser au programme qui l'a appelé la possibilité de remplir certaines fonctions à sa place, par exemple corriger les erreurs. Mais il doit toujours s'assurer que cela ne risque pas de perturber le fonctionnement des autres ressources

d'entrée-sortie. Connaissant toutes les demandes d'entrée-sortie sur les unités, il peut améliorer l'utilisation globale de l'ensemble des ressources d'entrée-sortie en exploitant les possibilités de fonctionnement simultané qui existent entre ces divers dispositifs : pour cela, il peut avoir à exécuter les demandes dans un ordre différent de leur ordre d'arrivée.

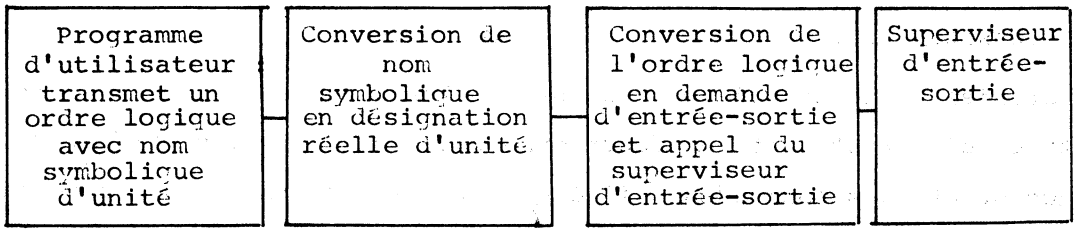


L'objet principal du travail qui va être décrit dans les chapîtres qui suivent est la conception et la réalisation d'un superviseur d'entrée-sortie pour le système à accès multiple GMS (Grenoble Monitor System), remplissant les fonctions énoncées ci-dessus. Mais l'appel à un superviseur d'entrée-sortie ne suffit pas à délivrer les programmeurs de tout problème concernant les entrées-sorties. Très souvent, un ensemble de programmes servent d'intermédiaires entre superviseur et utilisateur, éloignant ce dernier des contraintes du superviseur d'entrée-sortie. Le programme qui appelle le superviseur d'entrée-sortie doit construire la suite des commandes qui réalisera l'opération qu'il désire : ce peut être une tâche complexe car cela nécessite une connaissance approfondie du fonctionnement des matériels. Ce travail est d'autant plus délicat que deux unités de type différent ont un fonctionnement souvent très différent, et acceptent des commandes différentes. Le programme qui appelle le superviseur d'entrée-sortie doit de plus fournir l'interface d'appel

attendu par ce superviseur. Cet interface est généralement complexe car il contient, outre la suite des commandes à exécuter des renseignements concernant l'unité, les traitements d'erreurs, la synchronisation, etc... Ces problèmes sont évités quand le programme utilise des ordres logiques, et appelle le superviseur d'entrée-sortie par un intermédiaire. Cet intermédiaire traduit les ordres logiques en demandes d'entrée-sortie. Un ordre logique est, par exemple,

LIRE l'enregistrement A ou ECRIRE l'enregistrement A

A permettant de déterminer l'emplacement de l'enregistrement sur l'unité. L'utilisation d'un ordre logique peut ainsi délivrer le programmeur du souci de synchronisation, et de la nécessité de connaître le fonctionnement de l'unité. Le superviseur d'entrée-sortie présente une autre contrainte pour celui qui l'appelle : il faut spécifier l'unité sur laquelle doit être exécutée l'entrée-sortie. Le programme appelant doit être modifié chaque fois que l'unité d'entrée-sortie change. Pour supprimer cette contrainte, les unités sont désignées dans les programmes par des noms symboliques : un programme qui veut réaliser une entrée-sortie appelle un programme intermédiaire qui fait la conversion nom symbolique-désignation réelle de l'unité, grâce à une table de correspondance. Cette table est initialisée avant chaque exécution du programme.

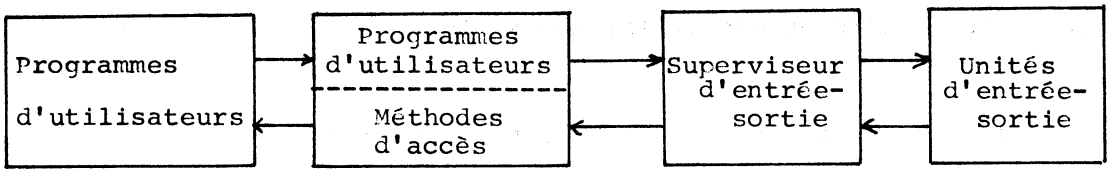


Grâce à ces facilités, le programmeur est plus indépendant des unités. Cependant, il lui faut encore connaître des informations sur ces unités : savoir sur quelle unité réside un enregistrement, et à quel endroit sur cette unité. Or ce qui l'intéresse généralement c'est de pouvoir accéder à des collections de données, sans se soucier de leur emplacement physique. Le système organise ces collections de données en fichiers, et gère lui-même les emplacements qu'il occupe. Il peut retrouver l'emplacement de ces fichiers par un catalogue. Les programmes permettant de créer ces fichiers et de lire leur contenu sont couramment appelés méthodes d'accès. Par ces méthodes d'accès le programme accède aux enregistrements sans connaître leur emplacement physique, en indiquant par exemple seulement un numéro d'enregistrement à l'intérieur de la collection. On retrouve à ce stade, au niveau global des fichiers, des problèmes qui se présentent au superviseur d'entrée-sortie. Les fichiers peuvent être d'accès limité, par exemple réservés à certains utilisateurs, ou accessibles seulement en lecture. Il peut y avoir aussi nécessité de synchroniser l'utilisation des fichiers, par exemple, empêcher la modification d'un fichier tant qu'il est exploité en lecture par des programmes d'utilisateurs. Tous ces concepts se retrouvent dans un certain nombre de systèmes actuels. Leur implantation peut être cependant très variable, selon les objectifs poursuivis. Aussi cette étude commence-t-elle par l'analyse des entrées- sorties dans quelques systèmes actuellement utilisés.

Le système CMS est un système conversationnel fonctionnant sur un ordinateur IBM 360 standard, ou sous le contrôle du système CP67, générateur de machines virtuelles. Les entrées-sorties de CMS sont un bon exemple de gestion simple d'entrées-sorties : pas de superviseur d'entrée-sortie mais un module différent pour chaque type d'unité. Les

mécanismes de synchronisation entre les entrées-sorties et les programmes sont très simplifiés.
Le système CP67 est beaucoup plus complet que CMS. Il contrôle un nombre d'unités assez variées. Outre les entrées-sorties qu'il effectue pour son propre compte, il simule pour les machines virtuelles qu'il génère toutes les entrées-sorties que celles-ci émettent. Ce système ne dispose pas, de même que CMS, d'un superviseur général d'entrée-sortie. Plusieurs modules totalement indépendants effectuent les entrées-sorties suivant les types d'unité. L'aspect le plus intéressant de CP67 en ce qui concerne les entrées-sorties est la simulation des entrées-sorties virtuelles sur des unités réelles. Cela pose des problèmes de correspondance entre les commandes virtuelles et les commandes réelles, entre les adresses virtuelles et les adresses réelles. On a aussi des problèmes de synchronisation : CP doit simuler les évènements attendus par la machine virtuelle dans l'ordre où ils se produiraient sur une machine réelle, et réactiver les machines virtuelles en attente d'un évènement lorsque cet évènement se produit.

Le système OS/360 offre un bon exemple de système où les entrées-sorties utilisent les concepts présentés précédemment. Le superviseur d'entrée-sortie de ce système gère toutes les entrées-sorties sur les unités. Un programme peut effectuer ses entrées-sorties en s'adressant directement au superviseur d'entrée-sortie (par l'interface EXCP), ou en passant par l'intermédiaire d'une méthode d'accès. La synchronisation entre une opération d'entrée-sortie et la tâche qui l'a émise se fait simplement par des fonctions sur évènements.



Entrées-sorties dans OS/360

MULTICS est un système conçu pour permettre à un grand nombre d'utilisateurs l'accès simultané à d'importantes quantités d'informations. Une des caractéristiques principales du système d'entrée-sortie de MULTICS est la facilité qu'ont les utilisateurs de désigner des collections d'informations par des noms symboliques. La liaison entre noms symboliques et collections d'informations peut être modifiée pendant l'exécution des programmes et laisse une très grande souplesse aux utilisateurs.

ESOPE, système développé pour fonctionner sur un ordinateur CII 10070, se caractérise, du point de vue des entrées-sorties demandées par un utilisateur, par une unification des mécanismes d'échange d'informations : mis à part les opérations sur les terminaux les échanges d'informations se font sur la base de fichiers qui transitent par la mémoire virtuelle. Les fichiers peuvent se trouver aussi bien en mémoire principale qu'en mémoire auxiliaire. L'utilisateur n'a pas accès directement aux unités d'entrée-sortie.

Après avoir analysé les points particuliers relatifs aux entrées-sorties dans ces systèmes, on analyse le superviseur d'entrée-sortie du système GMS. Ce système permet à plusieurs utilisateurs d'avoir accès aux ressources d'un ordinateur IBM360 depuis un terminal. Le superviseur d'entrée-sortie gère toutes les

ressources matérielles d'entrée-sortie selon les ordres qu'il reçoit du reste du système ou des utilisateurs. Les problèmes suivants y sont traités :

-Représentation et traitement des demandes d'entrée-sortie : cheminement dans les files d'attente, vérifications..

-Gestion optimale des unités d'entrée-sortie et des chemins qui y mènent.

-Synchronisation des entrées-sorties avec les programmes appelant le superviseur d'entrée-sortie.

Les derniers paragraphes de ce chapître montreront la progression des demandes d'entrée-sortie depuis l'utilisateur qui les émet jusqu'au superviseur d'entrée-sortie à travers les différents niveaux du système.

Le dernier chapître sera consacré à quelques problèmes n'ayant pas leur place dans l'étude du superviseur de GMS, qui est une réalisation particulière : problèmes posés par la pagination, évolution possible des entrées-sorties avec l'apparition de nouveaux matériels.

CHAPITRE 1

ENTREES-SORTIES

DANS LES SYSTEMES ACTUELS

Dans ce chapître nous étudions les entrées-sorties dans quatre systèmes fonctionnant à l'heure actuelle soit en exploitation quotidienne, soit de manière expérimentale : CP/CMS, OS/360, MULTICS, ESOPE. Les systèmes étudiés fonctionnent tous sur des modèles d'ordinateurs pour lesquels les unités centrales de traitement peuvent travailler en simultanéité avec les unités d'entrée-sortie. Une opération d'entrée-sortie est initialisée par une instruction d'une unité centrale ; la fin de l'opération est signalée à l'unité centrale par une interruption d'entrée-sortie. C'est un mécanisme "hardware" permettant aux unités d'entrée-sortie de stopper le déroulement des programmes et de faire traiter la fin d'entrée-sortie par l'unité centrale.

1.1 Déroulement d'une entrée-sortie sur IBM/360 (réf IBM360)

Le fonctionnement détaillé des entrées-sorties dans le système 360 est expliqué en annexe. Trois des systèmes étudiés dans ce chapître (CP67, CMS, OS/360) et le système GMS étudié au chapître 2 étant implantés sur des ordinateurs 360, nous rappelons ici brièvement le déroulement des entrées-sorties sur ces ordinateurs, ainsi que les principaux termes employés (figure 1.1).

Une unité d'entrée-sortie est reliée à la mémoire centrale par

l'intermédiaire d'un canal qui commande les transferts de données entre la mémoire centrale et les unités d'entrée-sortie. Un canal est généralement utilisé pour desservir plusieurs unités d'entrée-sortie. Il peut travailler simultanément avec l'unité centrale. Le canal exécute une suite d'instructions appelées commandes canal CCW (Channel command word). Chaque commande détermine une opération : lire un enregistrement, écrire un enregistrement, ... Une suite de commandes est appelée programme canal. L'adresse de la première commande à exécuter doit se trouver dans le mot d'adresse canal CAW (channel address word) lorsque l'opération d'entrée-sortie est initialisée. Le mot d'adresse canal se trouve à un emplacement fixe de mémoire centrale, à l'adresse 72. L'initialisation d'une opération d'entrée-sortie est faite par l'exécution de l'instruction SIO : cette instruction a un paramètre qui est l'adresse physique de l'unité d'entrée-sortie concernée. L'effet de SIO est d'activer le canal qui lit en mémoire centrale la première commande du programme canal. Dès qu'il a lu cette commande et vérifié sa validité, l'unité centrale reprend son activité. Le canal et l'unité centrale peuvent travailler simultanément pendant toute la durée de l'exécution du programme canal.

Le mot d'état programme PSW (Program Status Word) contient en permanence des indicateurs sur l'état de l'unité centrale, ainsi que l'adresse de la prochaine instruction à exécuter.

Le mot d'état canal CSW (Channel Status Word) indique dans quelles conditions s'est déroulée une entrée-sortie. Il est situé à un emplacement fixe en mémoire centrale.

La fin de l'exécution d'un programme canal est indiquée à l'unité centrale par une interruption d'entrée-sortie. Celle-ci se manifeste par la mise à jour du mot d'état canal, le rangement du mot d'état programme courant à une adresse fixe en mémoire centrale et le

chargement d'un nouveau mot d'état programme à partir d'une adresse fixe.

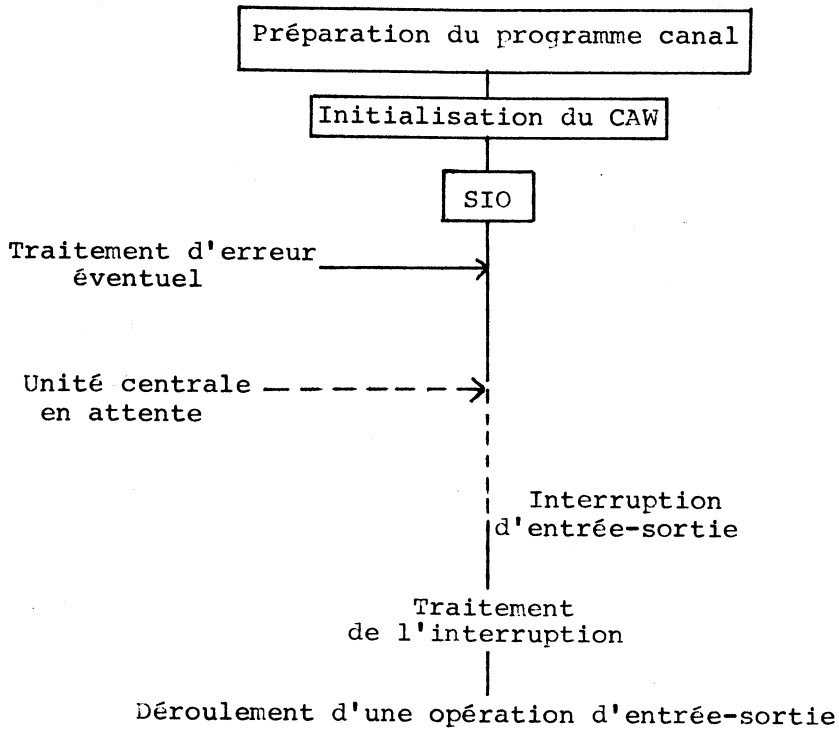


Figure 1.1

1.2 Les systèmes CP67/CMS (réf AH)

Les deux systèmes CP67 (Control Program 67) et CMS (Cambridge Monitor System) ont été développés pour fournir à plusieurs utilisateurs les ressources d'un ordinateur partagé. Le système CMS est un système conversationnel qui sert un seul utilisateur, par l'intermédiaire de la console de l'ordinateur. Le système CP67 est un système à partage de temps générateur de machines virtuelles : il simule, pour chaque utilisateur du système, un ordinateur 360 avec ses unités d'entrée-sortie. CMS tourne généralement sur une machine virtuelle générée par CP67.

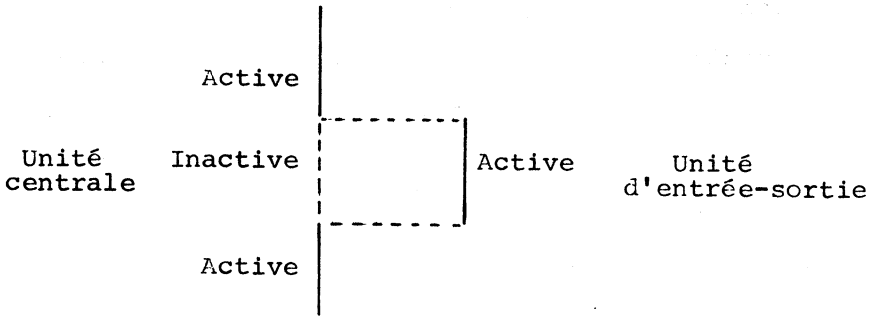
1.21 Entrées-sorties dans le système CMS (réf CP1, CMS)

CMS servant un seul utilisateur, les mécanismes d'allocation des ressources de la machine sont très simples. La configuration minimum sur laquelle on peut faire fonctionner CMS est la suivante :

- une console 1052
- un lecteur-perforateur de cartes et une imprimante
- un disque sur lequel sont conservés tous les fichiers du système
- un disque sur lequel l'utilisateur conserve ses fichiers personnels

L'utilisateur a toutes les unités de la configuration à sa disposition de manière permanente. Il n'y a donc pas eu besoin d'implanter des mécanismes de droits d'accès, permanents ou temporaires. La seule exception à cette règle a lieu pour l'utilisation du disque système : toute modification du contenu de ce disque est interdite à l'utilisateur. Comme ces unités de disque ne disposent pas d'un dispositif permanent pour empêcher les opérations d'écriture, cette protection doit être mise en place pour chaque opération d'entrée-sortie. C'est le rôle du système de faire cette protection.

Il n'y a pas, dans CMS, de superviseur d'entrée-sortie, c'est-à-dire de module par lequel passent toutes les demandes d'entrée-sortie. En effet, le système CMS est séquentiel. L'unité centrale est désactivée pendant le déroulement d'une entrée-sortie (figure 1.2). Il n'y a pas de partage de ressources d'entrée-sortie (à une exception près que nous verrons plus loin), et a fortiori pas besoin de l'allocateur de ressources que serait le superviseur d'entrée-sortie.



Fonctionnement des unités
pendant une entrée-sortie de CMS

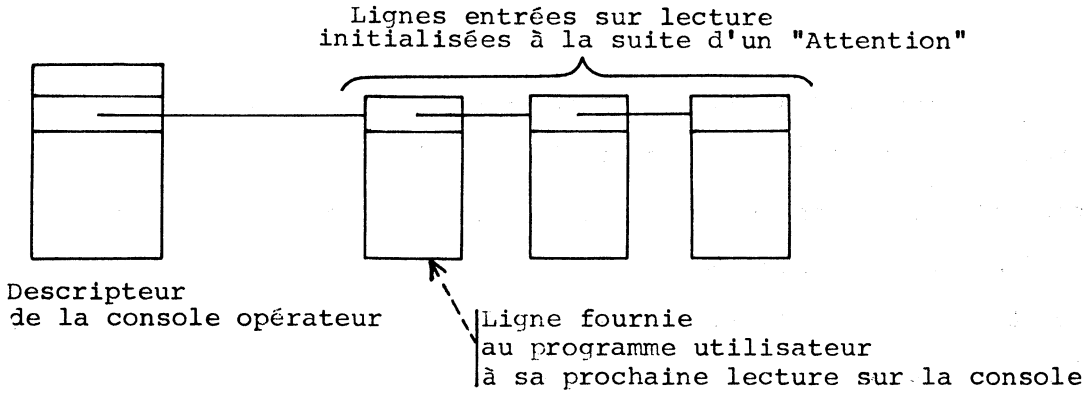
Figure 1.3

En effectuant ainsi ses entrées-sorties, le système ne tire aucun profit du parallélisme possible de fonctionnement entre l'unité centrale et les unités périphériques, et entre les unités périphériques elles-mêmes. C'est un désavantage quand le système fonctionne en "stand-alone", c'est-à-dire quand il contrôle directement tout le "hardware" ; mais quand il tourne sur une machine virtuelle générée par CP67, ce dernier profite de la désactivation d'une unité centrale virtuelle pour en activer une autre.

Il y a des exceptions à la règle du fonctionnement séquentiel des entrées-sorties. Les écritures sur la console opérateur se font sans que le programme qui les demande ne doive attendre la fin de l'entrée-sortie. En effet, ce sont souvent des messages ne nécessitant pas une réponse de l'opérateur (par exemple des diagnostics d'erreurs au cours d'une compilation). Pour pouvoir gérer ces entrées-sorties, le système représente chaque opération d'entrée-sortie sur la console par un descripteur. La console elle-même est décrite par une table, qui contient un pointeur sur la liste des opérations d'entrée-sortie en attente.

La deuxième exception à cette règle provient des entrées-sorties consécutives à une interruption "Attention". Cette interruption se

produit quand l'utilisateur appuie sur la touche "Attention" de la console 1052. Le système initialise alors une lecture à la console et relance le déroulement du programme en cours. L'utilisateur peut alors émettre des commandes pour contrôler le système (supprimer les écritures sur 1052, demander une nouvelle copie du système, ...), ou taper des lignes qui sont mises dans la file d'attente des lignes lues : il peut ainsi anticiper les demandes de lecture du système et supprimer les temps d'attente que cela entraîne (figure 1.3).



File d'attente des lignes lues à la console

Figure 1.3

1.22 Entrées-sorties dans le système CP67 (réf CP1, CP2)

Le système CP67 (Control Program 67) est un système générateur de machines virtuelles conçu pour fonctionner sur le modèle 67 de la série des ordinateurs IBM/360. En plus des dispositifs communs aux modèles de cette série, il dispose des mécanismes de pagination et de segmentation (réf IBM67).

L'espace adressable d'un utilisateur est l'ensemble des unités d'information qui peuvent être adressées par les instructions des programmes de l'utilisateur. Dans un ordinateur avec pagination, la

mémoire centrale est découpée en blocs de taille fixe (4096 octets sur le 360/67). L'espace adressable d'un utilisateur est découpé en blocs de la même taille, et ces blocs se trouvent soit en mémoire centrale, soit en mémoire auxiliaire. Les dispositifs cablés de segmentation et de pagination transforment les opérandes d'instructions qui contiennent des adresses de pages de l'espace adressable des utilisateurs, en adresses réelles des pages en mémoire centrale. Si un bloc de l'espace adressable ne se trouve pas en mémoire centrale lors de son utilisation, le système en est informé par une interruption : il doit alors faire venir cette page en mémoire centrale (réf PJD).

L'unité centrale travaille en mode standard lorsqu'elle n'utilise pas les dispositifs de pagination et de segmentation, et en mode étendu dans le cas contraire. Le mode de travail est indiqué par un bit du mot d'état programme.

Les canaux d'entrée-sortie n'ont pas accès aux dispositifs de pagination et de segmentation. Aussi toutes les commandes d'entrée-sortie doivent contenir les adresses réelles des données en mémoire centrale, et les commandes elles-mêmes doivent être en mémoire centrale quand l'opération d'entrée-sortie est initialisée.

CP étant un générateur de machines virtuelles, il doit fournir à chaque utilisateur les mêmes ressources que celles qu'il aurait à sa disposition s'il travaillait directement sur un ordinateur réel. Chaque machine virtuelle dispose de trois types de ressources :

- Une mémoire centrale. Dans ce but, on utilise les mécanismes de pagination et de segmentation.On peut ainsi faire coexister en mémoire centrale des parties quelconques d'espaces adressables de plusieurs utilisateurs.
- Une unité centrale de traitement. Celle-ci est attribuée par allocation de tranches de temps de l'unité centrale réelle. Les instructions privilégiées permettant de modifier l'état de l'unité

centrale, les clés de protection mémoire et les instructions d'entrée-sortie ne peuvent être utilisées librement par les machines virtuelles. Aussi, dès que l'unité centrale est allouée à une machine virtuelle, son fonctionnement se fait en mode problème. La tentative d'exécution des instructions privilégiées crée une interruption. Le système CP prend alors le contrôle et simule l'instruction.

- Les ressources d'entrée-sortie. Une machine virtuelle doit pouvoir utiliser des unités d'entrée-sortie de la même façon qu'une unité réelle. Une unité virtuelle peut être simulée par une unité réelle ou par un support de données disposant des mêmes fonctions de base, par exemple un fichier de "spooling".

Chaque utilisateur est représenté dans le système par un descripteur UTABLE. Celui-ci contient notamment l'identification de la machine virtuelle et le mot d'état programme courant de la machine virtuelle.

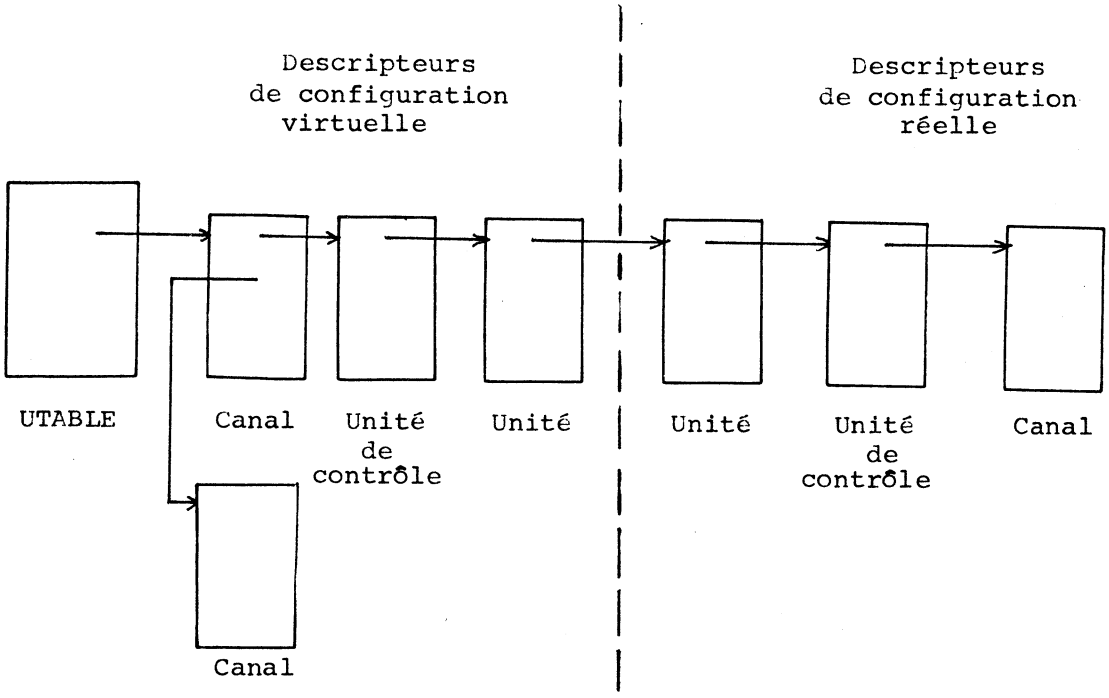
Nous allons étudier la simulation des entrées-sorties sur des disques virtuels, car cette simulation représente bien les problèmes créés par la correspondance virtuel-réel (réf LH). En effet, CP simule une unité de disque virtuelle par une unité réelle de même type, donc acceptant les mêmes commandes, ou par un ensemble de cylindres contigüs sur une unité de disque réelle de même type (cet ensemble est appelé mini-disque). On peut ainsi représenter plusieurs disques virtuels sur un seul disque réel, car un disque est une unité à accès direct.

Deux éléments importants diffèrent entre un mini-disque réel et un mini-disque virtuel :

- L'adresse physique de l'unité sur laquelle se trouve le mini-disque réel est généralement différente de l'adresse physique du mini-disque telle qu'elle est connue par la machine virtuelle.

- Le numéro du premier cylindre est toujours 0 sur le mini-disque virtuel ; le numéro du cylindre correspondant du mini-disque réel dépend de sa position sur le disque réel.

Les unités d'entrée-sortie virtuelles et réelles sont représentées dans le système par des tables. La figure 1.4 montre les liens entre ces différentes tables.



Liaison des descripteurs d'unité

Figure 1.4

Un programme tournant sur une machine virtuelle initialise une opération d'entrée-sortie par une instruction SIO et se met ensuite en attente d'interruption de fin d'entrée-sortie. La simulation des opérations d'entrée-sortie se fait donc en deux temps :

1-Transformation du SIO virtuel en SIO réel.

2-Transformations des interruptions réelles en interruptions virtuelles.

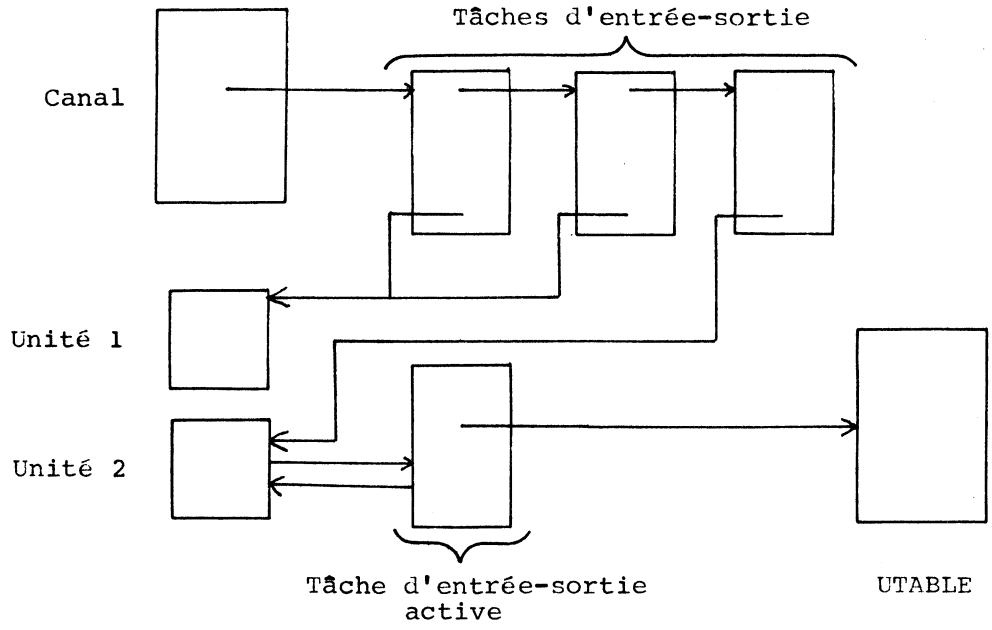
Quand une instruction SIO est exécutée par une machine virtuelle, elle est interceptée par CP car c'est une instruction privilégiée. Le module de traitement des interruptions programme détermine que l'interruption est causée par une instruction SIO à simuler.Il appelle alors le module VIOEXEC qui est chargé de simuler toutes les opérations d'entrée-sortie sur les disques virtuels. Ce module recherche les adresses des descripteurs du canal et de l'unité de contrôle virtuels menant à l'unité adressée par le SIO, ainsi que le descripteur de l'unité virtuelle elle-même (figure 1.4). Il détermine ainsi si le chemin menant à l'unité est libre : il n'est pas libre si un de ces éléments est occupé ou contient une interruption pendante ; un tel état est indiqué dans le descripteur correspondant, et il ne permet pas l'initialisation correcte du SIO. Dans ce cas, VIOEXEC réfléchit la même condition que celle qui se produirait dans pareille circonstance sur des unités réelles (code condition 1 ou 2 dans le mot d'état programme). L'opération n'est pas initialisée, et le contrôle est rendu au "DISPATCHER" : ce module alloue l'unité centrale au système et aux utilisateurs.

Si le chemin menant à l'unité est libre, VIOEXEC construit un descripteur de tâche d'entrée-sortie : ce descripteur matérialise dans le système la demande d'entrée-sortie. Il y met l'adresse de la procédure qui traitera les interruptions dûes à cette entrée-sortie. Il doit y mettre aussi l'adresse du programme canal qui simulera le programme canal de la machine virtuelle. Les canaux d'entrée-sortie n'utilisant pas les mécanismes de pagination et de segmentation, le programme canal virtuel doit être compilé : les adresses virtuelles sont remplacées par les adresses réelles correspondantes, éventuellement après avoir amené en mémoire centrale les pages de mémoire virtuelle. Les pages contenant des données sont verrouillées

en mémoire pendant toute la durée de l'entrée-sortie. Si un bloc de données dépasse une frontière de page, les pages réelles qui le contiennent ne sont pas toujours contigües en mémoire centrale : la commande virtuelle de transfert de ce bloc de données doit être remplacée par une suite de commandes reliées par un chaînage de données et relatives chacune à une page ou une fraction de page. Tout est alors prêt pour essayer de lancer l'opération sur l'unité réelle. A partir du descripteur de l'unité virtuelle, un lien permet de connaître l'unité réelle qui lui correspond, ainsi que le canal et l'unité de contrôle qui la desservent. Si le canal et l'unité de contrôle sont libres, une instruction SIO est exécutée. Le code condition est réfléchi à l'utilisateur qui peut être réactivé par le "DISPATCHER". Si le code condition indique une initialisation correcte, la tâche d'entrée-sortie est retirée de la file d'attente du canal et mise dans la file d'attente de l'unité réelle : ainsi la file d'attente du canal ne contient que des tâches d'entrée-sortie non initialisées (figure 1.5).

Si le chemin menant à l'unité n'est pas libre, la tâche d'entrée-sortie reste dans la file d'attente du canal, mais l'utilisateur n'est pas réactivé. Pour pouvoir le réactiver, il faudrait connaître le code condition à lui réfléchir : ce code n'est connu qu'après l'exécution du SIO sur l'unité réelle.

Descripteurs



Files d'attente des tâches d'entrée-sortie

Figure 1.5

L'utilisateur réactivé continue l'exécution de son programme jusqu'à ce qu'il exécute une instruction LPSW par laquelle il se mettra en attente de l'interruption virtuelle d'entrée-sortie.

Quand une interruption d'entrée-sortie se produit pour une unité réelle, les seules indications connues sont l'adresse physique de l'unité qui interrompt et le mot d'état canal indiquant la fin de l'entrée-sortie. Par l'adresse de l'unité, on retrouve le descripteur de cette unité et le descripteur du canal qui effectuaient l'opération d'entrée-sortie. Par le descripteur d'unité, on retrouve la tâche d'entrée-sortie active sur cette unité, et de là, le descripteur UTABLE de l'utilisateur pour qui l'entrée-sortie se fait. Le contrôle peut alors être donné à la procédure de traitement d'interruption

spécifiée dans la tâche d'entrée-sortie. Connaissant l'utilisateur pour qui l'entrée-sortie s'effectuait, elle peut lui réfléchir l'interruption. Cela amène les actions suivantes :

-Traduire le mot d'état canal réel en un mot d'état canal correspondant au programme canal virtuel.

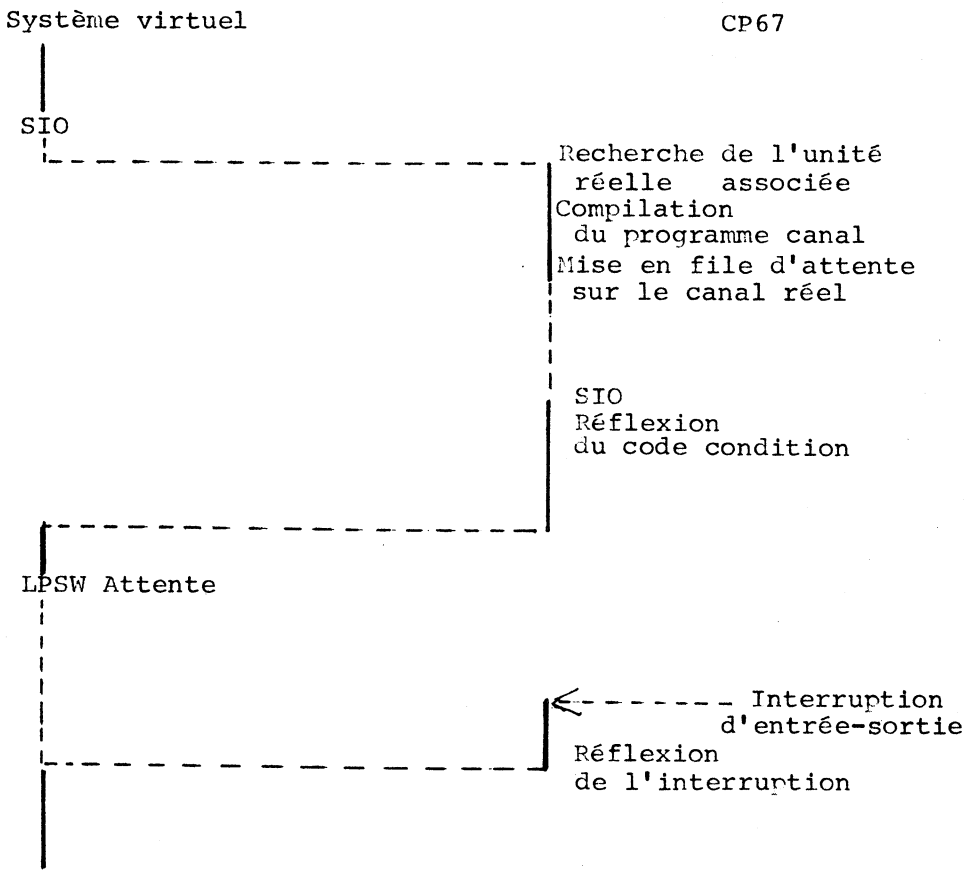
-Supprimer le verrouillage en mémoire centrale des pages contenant des données transférées.

-Mettre dans le descripteur UTABLE un indicateur PENDING à 1 : cette valeur indique au "DISPATCHER" qu'une interruption doit être réfléchie.

-Mettre à jour les indicateurs d'occupation du chemin virtuel menant à l'unité d'entrée-sortie virtuelle à laquelle se rapporte la tâche active.

Toutes ces actions étant faites, l'utilisateur peut être réactivé si un bit PENDING est à 1 et si le mot d'état programme dans la UTABLE autorise les interruptions pour ce canal. La simulation d'une interruption est faite ainsi :

- Transfert du PSW de la UTABLE dans l'ancien PSW d'entrée-sortie de la machine virtuelle.
- Chargement du PSW de la UTABLE avec le nouveau PSW d'entrée-sortie de la machine virtuelle.
- Transfert du mot d'état canal du descripteur de canal virtuel dans la mémoire de l'utilisateur.



Simulation d'une entrée-sortie sur disque

Figure 1.6

A ce moment, la simulation de l'opération d'entrée-sortie est terminée. Elle reflète fidèlement ce qui se serait passé sur une machine réelle.Il faut remarquer cependant qu'elle ne se fait pas au moindre coût, puisque les mêmes opérations sont effectuées pour le système virtuel et pour CP67 :

- Construire un programme canal
- Vérifier que le chemin menant à l'unité est libre
- Exécuter une instruction SIO
- Traiter les interruptions

Optimisation

CP ayant à sa charge la gestion de toutes les ressources d'entrée-sortie, il peut en optimiser l'utilisation. Par exemple, il effectue en deux temps une opération sur un disque nécessitant un déplacement du bras d'accès :

1 Déplacement du bras

2 Exécution de tout le programme canal

Dans le premier temps, le canal est occupé seulement au début de l'opération et CP67 peut ainsi faire chevaucher des opérations de positionnement de bras avec une entrée-sortie complète.

1.3 Entrées-sorties dans le système OS/360 (réf OSIOS, OSSPG)

Le système d'exploitation OS/360 est utilisé avec les ordinateurs IBM/360. C'est un système d'exploitation utilisant principalement le traitement par lots : un utilisateur soumet un travail au système par un paquet de cartes contenant des cartes de contrôle. Ces cartes identifient le travail, indiquent les programmes à exécuter et les ensembles de données sur lesquels ces programmes vont travailler. Les programmes qui composent le système d'exploitation sont divisés en programmes de traitement et programme de contrôle.

Le programme de contrôle a trois fonctions principales :

-lire et gérer les travaux

-superviser les unités de travail à effectuer

-s'occuper de la gestion physique des données

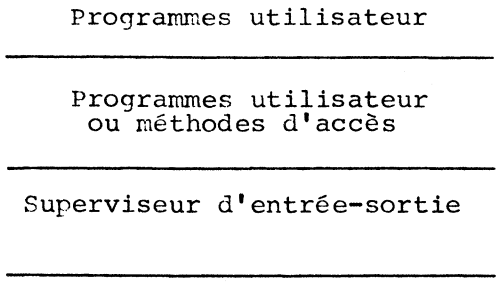
Les programmes de traitement comprennent les compilateurs, les programmes de service (par exemple l'Editeur de liens) et les programmes des utilisateurs.

Il existe trois programmes de contrôle ; le programme de contrôle choisi est fixé à la génération du système.

Le programme de contrôle PCP (Primary Control Program) amène en mémoire centrale et exécute une seule unité de travail à la fois.

Les programmes de contrôle MFT (Multiple Fixed Tasks) et MVT (Multiple Variable Tasks) permettent de réduire le temps d'attente de l'unité centrale. Plusieurs travaux se trouvent simultanément en mémoire centrale ; quand un de ces travaux est en attente d'un évènement, par exemple de fin d'entrée-sortie, un autre travail peut alors être activé : le système utilise la multi-programmation.

Quelque soit le programme de contrôle employé, le fonctionnement des entrées-sorties reste le même. Les opérations d'entrée-sortie demandées par les programmes des utilisateurs peuvent être demandées à deux niveaux : un niveau logique et un niveau physique. Réaliser une entrée-sortie à un niveau logique revient à appeler une fonction réalisant l'entrée-sortie (lecture, écriture) sans se préoccuper du déroulement physique de l'opération.



Les niveaux dans les entrées-sorties d'OS/360

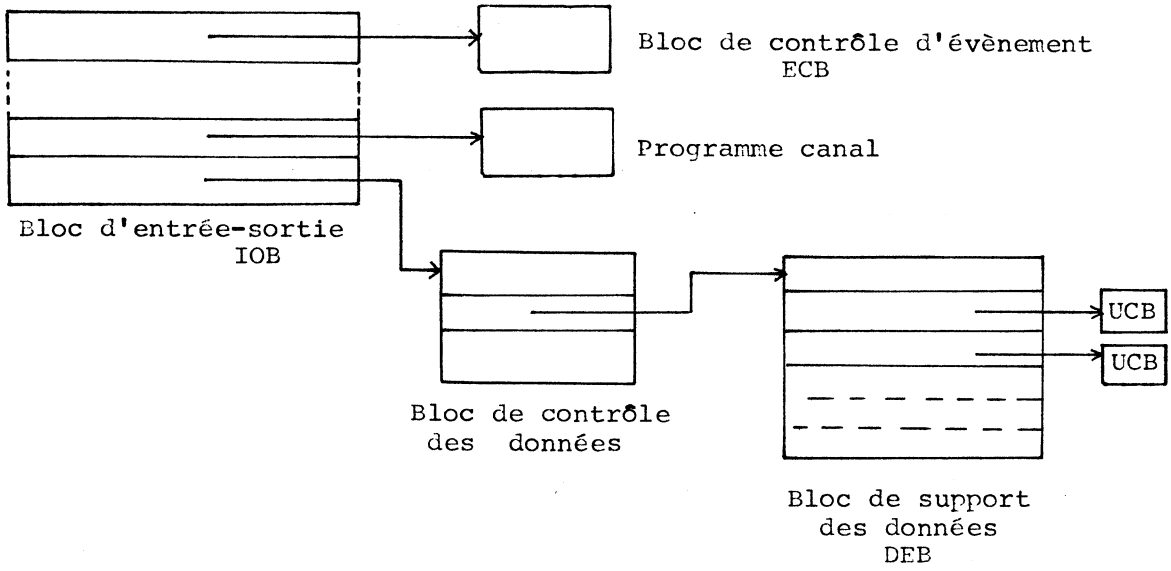
Figure 1.7

Toutes les opérations d'entrée-sortie sur les unités sont initialisées et traitées par un seul module, le superviseur d'entrée-sortie. Le superviseur d'entrée-sortie est appelé par la macro-instruction EXCP (EXecute Channel Program) qui génère une instruction SVC 0. Toute la gestion des chemins menant aux unités d'entrée-sortie lui est confiée, ces chemins pouvant être partagés par plusieurs travaux. Par contre, il ne se charge pas de décider quelles données ranger ou extraire de ces unités. Ce travail est laissé aux niveaux supérieurs.

1.31 Tables de contrôle

La communication entre le superviseur d'entrée-sortie et le programme d'application se fait par l'intermédiaire de tables de contrôle.En effet, dans un système tel que CMS (1.21), l'allocation des ressources d'entrée-sortie est permanente et l'utilisateur a toute liberté pour les utiliser comme il le désire, au besoin sans passer par les composants du système. Dans OS/360, la défection d'un travail ne doit pas affecter les autres travaux. Les tables de contrôle

permettent de normaliser le passage des informations et de faire des vérifications sur les accès demandés.



Liaison des tables de contrôle

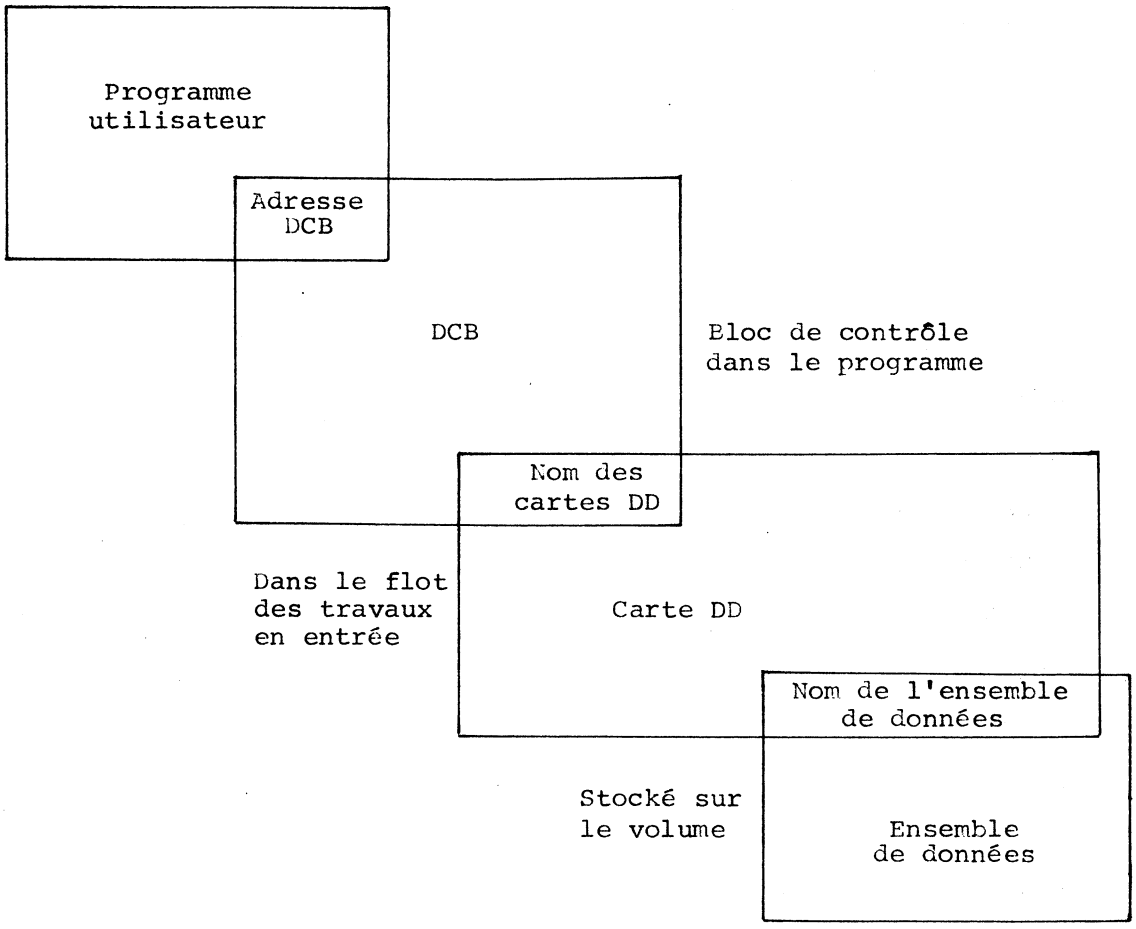
Figure 1.8

Le bloc de contrôle des données DCB (Data Control Block) est construit par les programmes qui appellent le superviseur d'entrée-sortie, directement ou indirectement. Il représente un ensemble de données sur lequel le programme peut travailler. Il permet de regrouper en un ensemble logique unique plusieurs fichiers différents. Le support physique des ensembles de données est retrouvé, au moment où le travail est soumis au système, par les cartes de contrôle de type DD (Data Definition). Un ensemble logique de données est défini par une suite de plusieurs cartes DD. Il a un nom qui se trouve sur la première carte DD ; ce nom est indiqué dans le DCB par le programmeur. On répond ainsi aux objectifs d'indépendance d'unité, puisque les ensembles de données utilisés ne sont définis qu'au moment de l'exécution du programme.

Le bloc de support des données DEB (Data Extent Block) contient une extension de l'information contenue dans le DCB. Il contient des informations sur les caractéristiques physiques de l'ensemble de données, et d'autres informations utilisées par le programme de contrôle. Par exemple, pour un ensemble de données sur disque, il contient les numéros des deux cylindres extrêmes alloués au fichier. Il contient aussi un pointeur vers le descripteur de l'unité UCB (Unit Control Block) qui supporte le fichier. Ces deux blocs ne sont pas modifiables par les programmes utilisateurs, car ils sont protégés par une clé différente de celle qu'utilisent ces programmes. Le superviseur d'entrée-sortie garde ainsi des informations toujours cohérentes sur les ressources physiques d'entrée-sortie. Le type d'accès autorisé (par exemple lecture seulement) peut être indiqué dans ces tables. Comme il ne peut les modifier, l'utilisateur ne peut outrepasser ses droits d'accès. Le bloc de support des données ne peut pas être créé avant la soumission du travail auquel il se rapporte, puisque l'ensemble de données est spécifié seulement à ce moment. Aussi tout programme utilisant un ensemble de données doit, avant d'appeler le superviseur d'entrée-sortie, appeler une procédure d'ouverture, en lui fournissant en paramètre l'adresse du bloc de contrôle des données. Elle peut disposer alors de tous les éléments pour faire la liaison entre le DCB et le support physique de l'ensemble des données. Elle construit un bloc de support des données et complète le bloc de contrôle des données grâce aux informations fournies par les cartes DD.

Le bloc d'entrée-sortie IOB (Input-Output Block) contient des informations relatives à une demande d'entrée-sortie ainsi qu'à la progression de l'opération d'entrée-sortie. L'adresse du bloc d'entrée-sortie est le seul paramètre à fournir à la macro-instruction EXCP, les adresses des autres tables étant contenues dans ce bloc.

Le bloc de contrôle d'évènement ECB (Event Control Block) est utilisé pour synchroniser l'opération d'entrée-sortie avec le déroulement du programme appelant.Il fournit d'autre part un code décrivant la façon dont s'est déroulé le programme canal.



Liaison programme utilisateur-ensemble de données

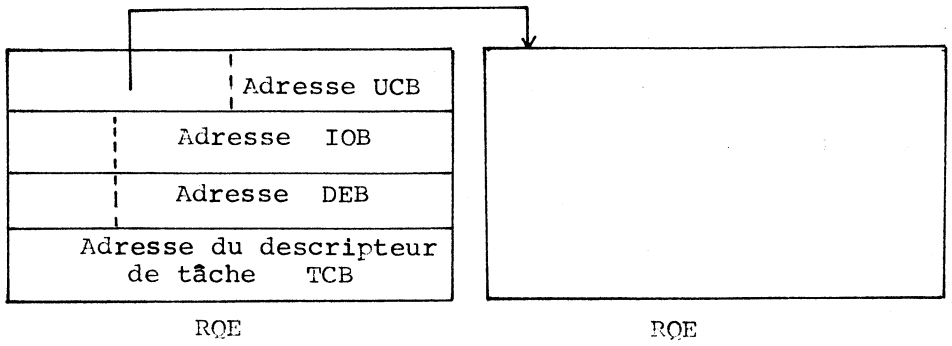
Figure 1.9

1.32 Logique du superviseur d'entrée-sortie

Le superviseur d'entrée-sortie a deux points d'entrée qui correspondent aux deux fonctions qu'il doit remplir. Le premier est utilisé par les programmes qui émettent la macro-instruction EXCP,

pour prendre en compte la demande d'entrée-sortie. Le deuxième est réservé au système : il est utilisé pour le traitement des interruptions d'entrée-sortie.

Au premier point d'entrée, le superviseur d'entrée-sortie vérifie la validité de l'information contenue dans les blocs de contrôle. Les trois blocs de contrôle ECB, DCB et IOB qui sont fournis par l'appelant doivent être dans la zone de mémoire qu'il peut modifier ; sinon, cela signifierait que ces blocs proviennent d'un autre travail, et il y a peu de chance qu'ils répondent à l'entrée-sortie demandée. Pour des raisons de protection, le bloc DEB doit être protégé par une clé nulle qui en réserve la modification au programme de contrôle. Si l'opération a lieu sur un disque, l'adresse de SEEK (numéro de cylindre, numéro de piste) spécifiée dans le bloc IOB doit être comprise dans l'espace alloué à l'ensemble de données et indiqué par le bloc DEB. Cette adresse est utilisée par le superviseur d'entrée-sortie pour positionner le bras du disque. Il assure ainsi que le programme canal spécifié par le bloc IOB n'accèdera pas à d'autres enregistrements que ceux de l'ensemble de données auquel se rapporte l'entrée-sortie. Si une de ces données est invalide, la demande d'entrée-sortie est refusée.
Sinon, le superviseur d'entrée-sortie construit un bloc de contrôle RQE (Request Queue Element) qui reflète la demande d'entrée-sortie. Cette requête (figure 1.10) interne au superviseur d'entrée-sortie parcourt des files d'attente qui reflètent la progression des demandes d'entrée-sortie.



Requête d'entrée-sortie

Figure 1.10

Le superviseur d'entrée-sortie lance la demande d'entrée-sortie si le canal et l'unité sont libres. Sinon, la requête est mise en file d'attente pour qu'elle soit exécutée ultérieurement quand le chemin sera libre. Le contrôle est immédiatement rendu au programme appelant.Si le chemin est libre, le superviseur d'entrée-sortie fait un traitement préliminaire pour certains types d'unités : par exemple, pour les unités à accès direct (disques), il lance une commande isolée de positionnement du bras d'accès. L'exécution de cette commande peut se faire simultanément avec d'autres opérations. Si le traitement préliminaire est achevé immédiatement, le système obtient l'adresse du programme canal dans le bloc IOB et émet l'instruction SIO par appel à un sous-programme commun à toutes les unités, et il traite les éventuelles erreurs indiquées par SIO. Le contrôle est ensuite rendu à l'appelant.

Lors d'une entrée sur interruption, le superviseur d'entrée-sortie regarde si l'interruption est associée à une requête. Si des conditions d'erreur sont indiquées par l'interruption, le superviseur d'entrée-sortie fait appel à la procédure appropriée dépendante de l'unité. Une nouvelle requête est alors choisie et démarrée sur le canal.

De manière habituelle, la communication entre le superviseur d'entrée-sortie et les programmes qui l'appèlent se fait par l'intermédiaire des tables de contrôle . Les erreurs survenant sur les unités sont traitées par des procédures connues du seul superviseur d'entrée-sortie. Cependant, certains utilisateurs peuvent vouloir effectuer leur propre traitement d'erreur ; il faut alors qu'ils spécifient les adresses de ces procédures dans le bloc DCB. Ces procédures "Appendage" peuvent recevoir le contrôle quand l'un des évènements suivants se produit :

- Avant l'exécution de l'instruction SIO
- Quand l'adresse de positionnement du bras spécifiée dans le bloc IOB pour un disque est hors des limites attribuées à l'ensemble des données
- Quand une interruption "Fin sur canal" est reçue.
- En cas de fin anormale d'entrée-sortie.

L'appelant peut ainsi faire effectuer des procédures autres que les procédures standard du superviseur d'entrée-sortie.

1.33 Synchronisation appelant-superviseur d'entrée-sortie

La synchronisation entre l'appelant et le superviseur d'entrée-sortie n'est pas laissée entièrement à la charge du superviseur d'entrée-sortie. Elle se fait par deux fonctions sur évènements ; un évènement est matérialisé dans le système par un bloc de contrôle d'évènement ECB. Le superviseur d'entrée-sortie rend le contrôle à l'appelant dès qu'il a pris en compte une demande d'entrée-sortie. Mais l'appelant peut vouloir attendre la fin de l'entrée-sortie avant de continuer son travail. Il émet alors un appel à la fonction WAIT du système en lui donnant l'adresse du bloc ECB

dont l'adresse doit être spécifiée dans le bloc IOB. La fonction WAIT met alors ce programme en attente. Quand l'entrée-sortie se termine, le superviseur d'entrée-sortie poste l'évènement décrit par le bloc ECB dont il trouve l'adresse dans le bloc IOB : il y met un code indiquant le résultat de l'entrée-sortie et il appelle la fonction POST du système qui réactive la tâche correspondant au programme appelant (figure 1.11). Le programme appelant le superviseur d'entrée-sortie doit donc effectuer lui-même la synchronisation avec l'entrée-sortie et ne peut laisser ce travail au superviseur d'entrée-sortie.

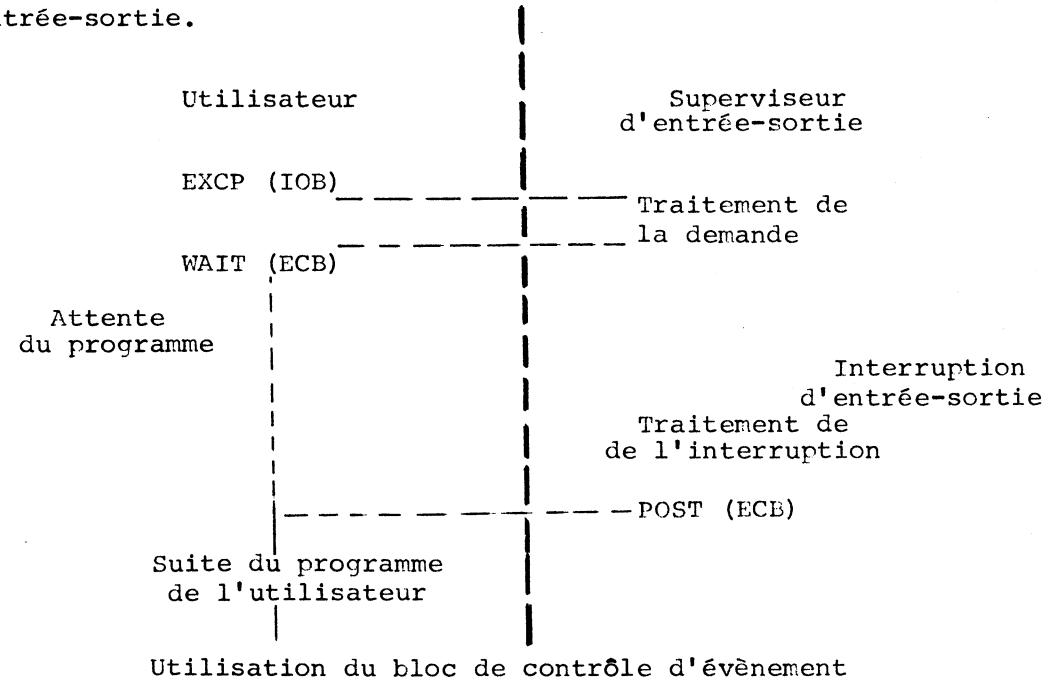


Figure 1.11

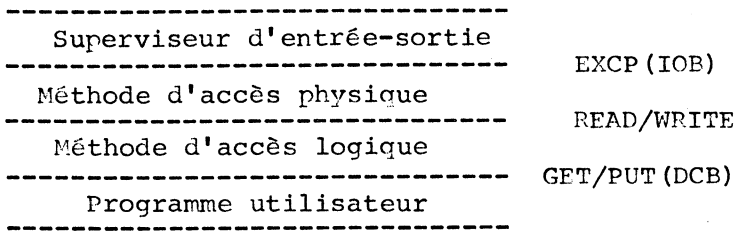
1.34 Méthodes d'accès

L'utilisation de la macro-instruction EXCP laisse malgré tout un gros travail au programmeur d'application qui veut faire des entrées-sorties, alors qu'il lui suffirait de disposer de fonctions simples de lecture et d'écriture. C'est dans ce but qu'on a inséré

entre le niveau "programmes d'application" et le niveau "superviseur d'entrée-sortie " un niveau "méthodes d'accès". Les méthodes d'accès constituent un ensemble de programmes qui sont employés par les programmes d'application par l'intermédiaire de fonctions logiques : lire un enregistrement, écrire un enregistrement, ... Les enregistrements logiques sont définis par le programme d'application qui doit fournir uniquement un bloc DCB, alors que les enregistrements physiques dépendent de l'unité. C'est ainsi que, par exemple, un enregistrement physique sur disque peut contenir plusieurs images de carte. Le programmeur d'application n'a plus à se préoccuper du fonctionnement des unités. De plus, le programme d'application utilisant des fonctions logiques, le même programme peut utiliser, dans des travaux successifs, des ensembles de données situés sur des unités de types différents. Il suffit qu'ils aient la même organisation logique, par exemple fichier séquentiel d'images de carte. L'ensemble de données sur lequel travaille le programme est toujours défini dans les cartes de contrôle.

Les méthodes d'accès elles-mêmes se présentent en deux niveaux qui se rapportent tous deux à des fichiers. Dans le niveau logique, l'accès aux fichiers se fait par les fonctions GET et PUT qui donnent l'accès à des enregistrements logiques. La méthode d'accès logique substitue l'appel à ces fonctions par un ou plusieurs appels à la méthode d'accès physique en fonction de l'organisation du fichier et de son support. L'appel à la méthode d'accès physique se fait par les fonctions READ ou WRITE. L'appel à une de ces fonctions permet de lire un enregistrement physique d'unité, sans avoir à en connaître le fonctionnement. C'est la méthode d'accès physique qui transforme cet appel en un appel au superviseur d'entrée-sortie, après avoir construit un bloc IOB. La méthode d'accès physique ne fournit pas de

synchronisation automatique entre la méthode d'accès logique et le déroulement de l'entrée-sortie. C'est la méthode d'accès logique qui doit faire explicitement cette synchronisation.



Niveaux d'exécution d'une entrée-sortie

Figure 1.12

Notons que l'utilisateur peut ne pas employer les méthodes d'accès logiques et physiques.Il peut appeler directement le superviseur d'entrée-sortie, ou appeler uniquement la méthode d'accès physique. Dans ces cas-là, la synchronisation avec l'entrée-sortie lui incombe toujours.

1.4 Entrées-sorties dans MULTICS (réf MU, OMD, FO)

Le système MULTICS (MUltiplexed Information and Computing System) doit permettre l'accès d'une quantité importante de données à plusieurs utilisateurs simultanément, selon les techniques les plus avancées. Il a été mis au point sur un ordinateur GE645 disposant de la pagination et de la segmentation.

1.41 Présentation générale des entrées-sorties

Un des buts recherchés pour les entrées-sorties était une grande facilité d'emploi par les utilisateurs et les programmeurs système, ainsi que la possibilité de disposer d'outils adaptés aux problèmes à

résoudre. Ce système propose aux utilisateurs une gestion de fichiers basée sur la segmentation. Les fichiers sont placés dans des segments, et les éléments des fichiers peuvent être adressés directement par l'unité centrale. Les fichiers sont conservés avec le format de page en mémoire centrale ou auxiliaire : l'accès à un de leurs éléments entraîne un défaut de page, si cet élément n'a pas été amené précédemment en mémoire centrale. Les transferts entre mémoire centrale et mémoire secondaire sont laissés entièrement à la charge du système et sont complètement transparents à l'utilisateur. Celui-ci est donc délivré du souci d'exécuter des ordres d'entrée-sortie, que ce soit à un niveau logique ou à un niveau physique. Ce mode d'accès est suffisant pour la plupart des utilisateurs, et il facilite le travail du système puisque toutes les données sont enregistrées sous un format unique.

Toutefois, cette solution ne résoud pas tous les problèmes des entrées-sorties dans le système. D'une part, les transferts de page entre mémoire centrale et mémoire auxiliaire se font par des opérations d'entrée-sortie physiques. D'autre part, il faut pouvoir aussi accéder aux unités d'entrée-sortie qui ne sont pas exclusivement réservées à la pagination et aux fichiers, et pour lesquels des ordres explicites d'entrée-sortie sont émis par des modules du système ou des utilisateurs. Pour cela, l'utilisateur doit pouvoir faire exécuter des entrées-sorties sur mesure, c'est-à-dire ne disposer que des fonctions d'entrée-sortie qui lui sont nécessaires. Il peut utiliser des fonctions indépendantes de l'unité sur laquelle il veut effectuer une entrée-sortie, par exemple

"écrire" sur mon unité "TOTO" la zone de 80 caractères qui se trouve à l'adresse 10000.

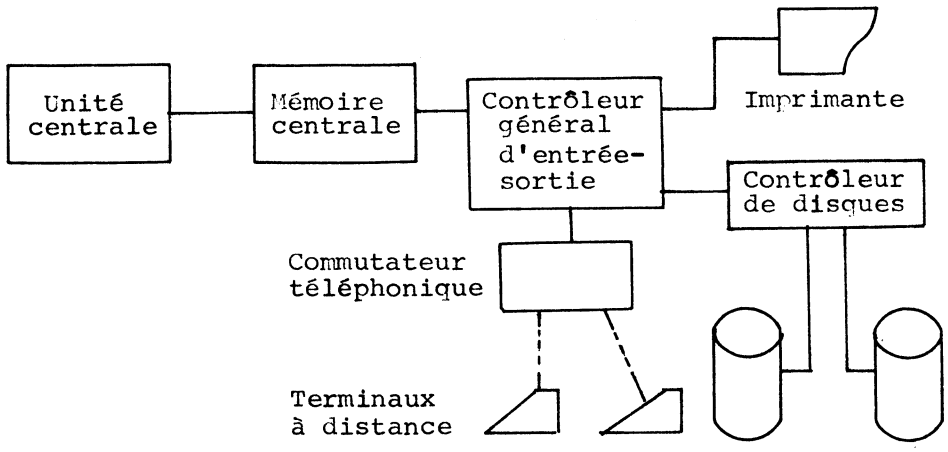
L'unité elle-même, comme on le voit sur cet exemple, peut être désignée par un nom symbolique. Ceci permet d'exprimer de manière unique dans un programme une opération pouvant s'effectuer sur plusieurs unités différentes, et de déterminer dynamiquement à l'exécution sur quelle unité travailler.

Mais un utilisateur peut aussi avoir le contrôle complet d'une unité d'entrée-sortie si l'administrateur du système lui en donne le droit. Il doit alors connaître le fonctionnement de l'unité, et il construit lui-même les séquences d'ordre qui sont directement transmises à l'unité. Pour remplir toutes ces fonctions, un système d'entrée-sortie IOCS (Input-Output Control System) a été conçu. Ses deux rôles principaux sont :

- La gestion physique des unités d'entrée-sortie
- La liaison des unités d'entrée-sortie avec les noms symboliques utilisés pour les représenter.

1.42 Gestion physique des unités d'entrée-sortie

Toutes les unités d'entrée-sortie sont reliées à la mémoire centrale par l'intermédiaire d'un contrôleur général d'entrée-sortie (réf OMD). Il reçoit les ordres d'entrée-sortie émis par les programmes en mémoire centrale, et il les transmet aux unités. Quand les unités ont terminé les opérations qu'elles effectuaient, elles le signalent au contrôleur d'entrée-sortie qui émet des demandes d'interruption de l'unité centrale, après avoir établi des priorités entre ces interruptions.



Squelette de configuration supportant MULTICS

Figure 1.13

Le module d'interface du contrôleur général d'entrée-sortie GIM (General Interface Module) est responsable de toute la gestion de ce contrôleur et des unités d'entrée-sortie (figure 1.14). Ses fonctions sont donc l'initialisation des opérations d'entrée-sortie, le traitement des interruptions, la reconnaissance de l'achèvement d'une tâche d'entrée-sortie, et la transmission de l'information d'état fournie par le contrôleur général d'entrée-sortie.

Les opérations d'entrée-sortie peuvent se dérouler de manière synchrone ou asynchrone, et l'utilisateur peut spécifier le mode qu'il désire. Quand une opération se déroule de manière synchrone, le contrôle n'est rendu à l'appelant que lorsque l'opération est achevée. C'est une autre fonction du GIM que de faire réactiver un processus en attente : pour cela, il appelle le contrôleur du système qui réveille le processus suspendu.

1.43 Liaison noms symboliques-unités d'entrée-sortie

La liaison entre les unités d'entrée-sortie et les noms symboliques employés par les programmeurs est faite par une table de liaison. Une entrée dans cette table est créée par un appel à la fonction ATTACH :

call ATTACH ("nom symbolique", DIM, "nom d'unité")

"nom symbolique" est désormais associé à "nom d'unité". Le module d'interface d'unité DIM (Device Interface Module) est un module spécialisé pour un type d'unité. Il est chargé de transformer les appels aux fonctions de lecture ou d'écriture sur l'unité associée au nom symbolique en une requête d'entrée-sortie assimilable par l'unité. Il doit donc construire un programme canal, et peut faire un traitement propre à l'unité, par exemple pour un terminal, convertir le code utilisé par le système en code employé par l'unité. Quand la requête est prête, il appelle le module GIM. Il peut communiquer avec lui jusqu'à ce que l'entrée-sortie soit achevée (figure 1.14) .

Une opération d'entrée ou de sortie demandée par un utilisateur sur une unité associée à un nom symbolique peut donc se faire comme suit :

CALL READ ("nom symbolique", "zone")

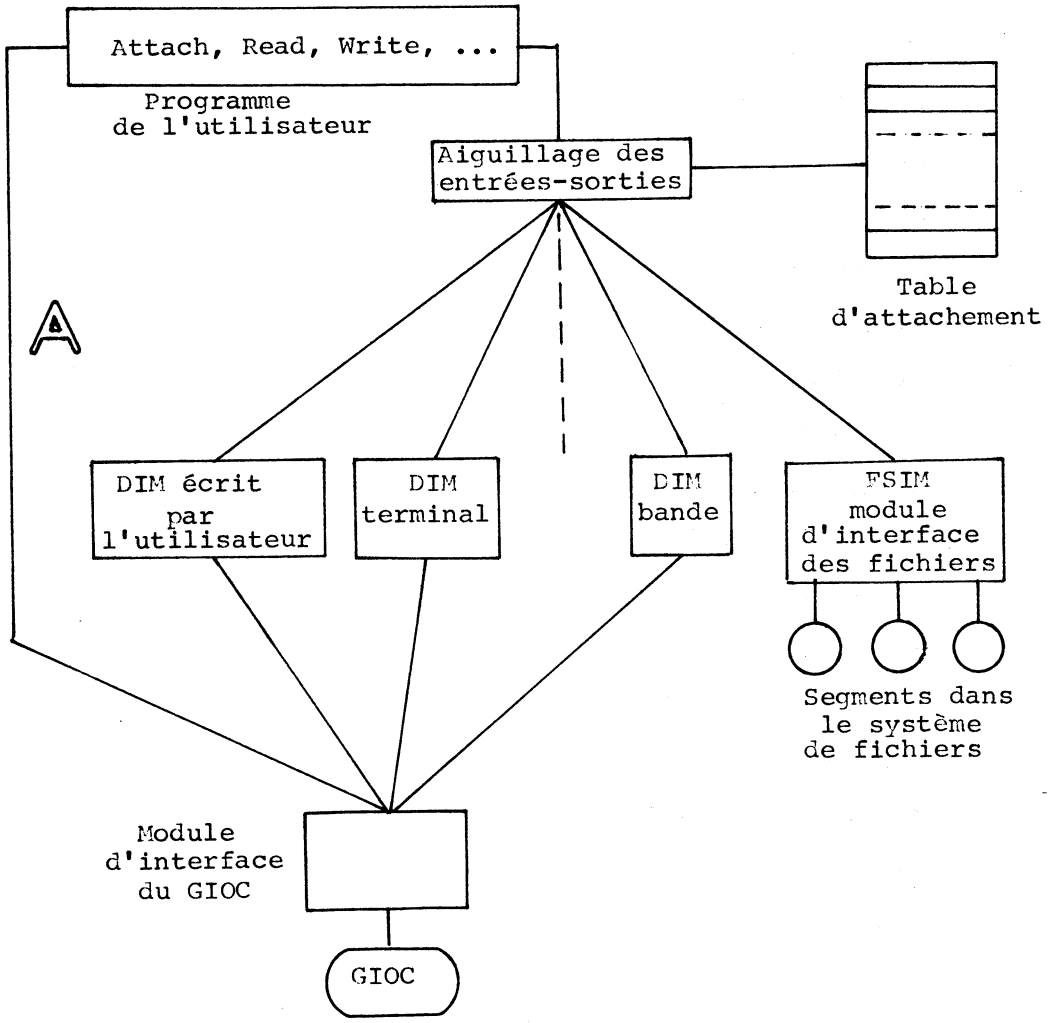
ou CALL WRITE("nom symbolique", "zone")

Cet appel est reçu par le module de contrôle des entrées-sorties (encore appelé aiguilleur d'entrée-sortie) : il explore la table d'attachement pour connaître l'unité réelle associée au nom symbolique, et vérifie que les paramètres de l'appel sont compatibles avec les informations contenues dans la table d'attachement. Il fait alors appel au module spécialisé DIM indiqué dans cette table.

Une unité peut très facilement être réservée ou gérée par un utilisateur dans le système.Il peut gérer ses entrées-sorties de deux manières :

1 Il fait appel directement au module d'interface GIM (lien A sur la figure 1.14) ; dans ce cas, il est totalement dépendant de l'unité, et ne peut pas profiter des possibilités de désignation symbolique des unités.

2 Il écrit un module DIM pour cette unité. Il dispose alors de toutes les facilités apportées par l'adressage symbolique des unités.



Organisation du système d'entrée-sortie

Figure 1.14

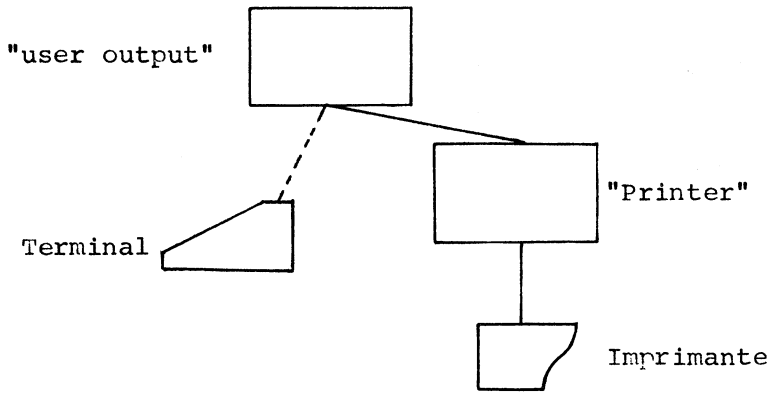
Les modules DIM sont les seuls modules standard du système à connaître réellement les unités physiques. Leur rôle est de transférer des données dans ou à partir de la mémoire centrale. Mais il peut également exister des modules appelés pseudo-DIMs qui ne travaillent pas directement sur des unités d'entrée-sortie.

Deux de ces pseudo-DIMs jouent un très grand rôle dans le système :

1 Le module d'interface du système de fichiers FSIM (File System Interface Module) permet de traiter un segment dans le système de fichiers comme une unité d'entrée-sortie. Si un segment est attaché à un nom symbolique par la fonction ATTACH, un ordre d'écriture ou de lecture sur l'unité désignée par ce nom symbolique se traduira en fait par une écriture ou une lecture dans le segment.
Ex. CALL ATTACH ("fichier sortie", FSIM, "nom de segment")
Toutes les sorties dans l'unité associée au nom symbolique "fichier sortie" seront rangées dans le segment "nom de segment".

2 Le module SYNONYM permet d'associer à un nom symbolique un autre nom symbolique qui correspond lui-même à une unité. Une opération s'adressant à l'unité désignée par le premier nom est dirigée vers l'unité qui correspond au second nom. Ceci offre un moyen très simple de changer dynamiquement l'unité avec laquelle un programme communique. Supposons par exemple qu'un utilisateur s'aperçoive que les informations qu'il fait sortir sur son terminal sont trop longues et qu'il veuille les diriger vers une imprimante rapide (figure 1.15). Soit "user output" le nom symbolique du terminal en sortie, et "printer" le nom symbolique de l'imprimante. Il suffit alors de faire les appels suivants :
CALL ATTACH ("printer", DIM, numéro d'imprimante)
CALL DETACH ("user output")

CALL ATTACH ("user output", SYNONYM, "printer")



Changement d'affectation d'un nom symbolique

Figure 1.15

Le premier appel crée dans la table d'attachement une connexion entre le nom symbolique "printer" et une imprimante réelle de la configuration. Le deuxième appel supprime la connexion entre le nom symbolique "user output" et l'unité qui lui correspondait, c'est-à-dire le terminal pour les sorties. Le troisième appel associe le nom symbolique "user output" à l'imprimante par l'intermédiaire du nom symbolique "printer". Désormais, toutes les sorties envoyées sur l'unité désignée par "user output" seront aiguillées vers l'imprimante désignée par "printer".

L'utilisation conjuguée du module SYNONYM et du module FSIM permet d'avoir une très grande souplesse d'emploi des unités et des fichiers. Une application très intéressante est celle qui consiste à associer deux segments aux noms symboliques désignant un terminal, en entrée et en sortie. Dans ce cas, la communication directe avec l'utilisateur est supprimée. Ce type de liaison est utilisé pour faire exécuter un travail de fond ("batch processing") sans modification du programme ni des ordres de contrôle.

1.5 Entrées-sorties dans le système ESOPE (réf ESO, CK, SK)

Le système ESOPE est un système réalisé en France, à l'Institut de Recherche en Informatique et Automatique (IRIA). Il est essentiellement conversationnel, l'accès à ce système se faisant par des terminaux (machines à écrire, consoles graphiques). Il tourne sur un ordinateur CII 10070. La mémoire principale est divisée en pages de 512 mots de 32 bits. Un mécanisme cablé de traduction dynamique des adresses, la mémoire topographique, permet d'associer à chaque utilisateur une mémoire virtuelle de 256 pages au maximum. La mémoire topographique est constituée de 256 registres et elle est adressée par le numéro de page virtuelle. Le contenu de chaque registre indique la page physique associée. En outre, un indicateur de protection est associé à chacun de ces registres et utilisable seulement quand l'unité centrale travaille en mode esclave. Il indique l'accès autorisé à la page virtuelle :

- Tout accès permis
- Ecriture interdite
- Lecture seule permise
- Tout accès interdit

Le dernier type d'accès est utilisé pour indiquer l'absence de page physique ou l'inexistence de page virtuelle (cas d'une mémoire virtuelle ayant moins de 256 pages).

Un ensemble de disques à têtes fixes RAD est utilisé pour conserver les fichiers des utilisateurs et les pages virtuelles des utilisateurs qui ne peuvent être en mémoire centrale. La configuration peut comprendre aussi des terminaux des périphériques lents classiques : lecteur de cartes, imprimante, dérouleurs de bande. Nous étudions ici deux points importants relatifs aux entrées-sorties :

- La gestion et l'accès aux fichiers permanents des utilisateurs

- La gestion des périphériques classiques (lecteur de cartes, imprimante)

1.51 Accès aux fichiers des utilisateurs

On retrouve dans ESOPE une idée qui avait déjà été exploitée dans MULTICS, à savoir la correspondance entre un fichier et la mémoire virtuelle d'un utilisateur. Toutefois, pour MULTICS, le "hardware" permet d'associer un fichier complet à un segment. L'absence de segmentation sur le 10070 contraint à se limiter à une association page de mémoire virtuelle - article de fichier. Pour cela, les fichiers rangés sur les disques à tête fixe sont découpés en articles ayant la taille d'une page de mémoire virtuelle. Un article de fichier possède à un instant donné un seul support, soit sur mémoire auxiliaire, soit en mémoire centrale. La mémoire virtuelle est un sous-ensemble de l'espace virtuel, ou espace des fichiers. La localisation d'un article de fichier est obtenue par l'utilisation des tables suivantes :

Table des fichiers opérationnels TFO

Une entrée de cette table correspond à un fichier dont les articles sont situés en mémoire principale ou sur RAD. Ces fichiers reçoivent à leur création un numéro interne f qui sert d'index dans la TFO. Une entrée de cette table contient en particulier un pointeur vers la table TIF décrivant l'implantation du fichier en mémoire.

Table d'implantation de fichier TIF

Cette table contient pour chaque article du fichier qu'elle décrit :

- l'emplacement original de l'article sur le disque RAD
- le numéro r de la page physique dans laquelle se trouve l'article (r est nul si l'article ne se trouve pas en mémoire centrale).

Le numéro d'article sert d'index d'entrée dans cette table.

Table des pages virtuelles TPV

Cette table est associée à chaque mémoire virtuelle. Elle définit la composition de la mémoire virtuelle en articles de fichier. Le numéro de la page virtuelle sert d'index d'entrée dans cette table. Une entrée contient en particulier :

- le numéro de fichier f et le numéro de l'article couplé à la page virtuelle (f=0 indique l'absence de couplage)
- le numéro de la page physique correspondant à la page virtuelle, utilisé pour le chargement de la mémoire topographique.

Le mécanisme d'accès à l'information est indiqué à la figure 1.16. L'application qui permet, à partir d'une page de mémoire virtuelle, de retrouver le support physique de la page correspondante est consécutive à l'exécution des deux fonctions suivantes :

couplage : associe une page de mémoire virtuelle à un fichier

implantation de fichier : indique le support physique d'un fichier

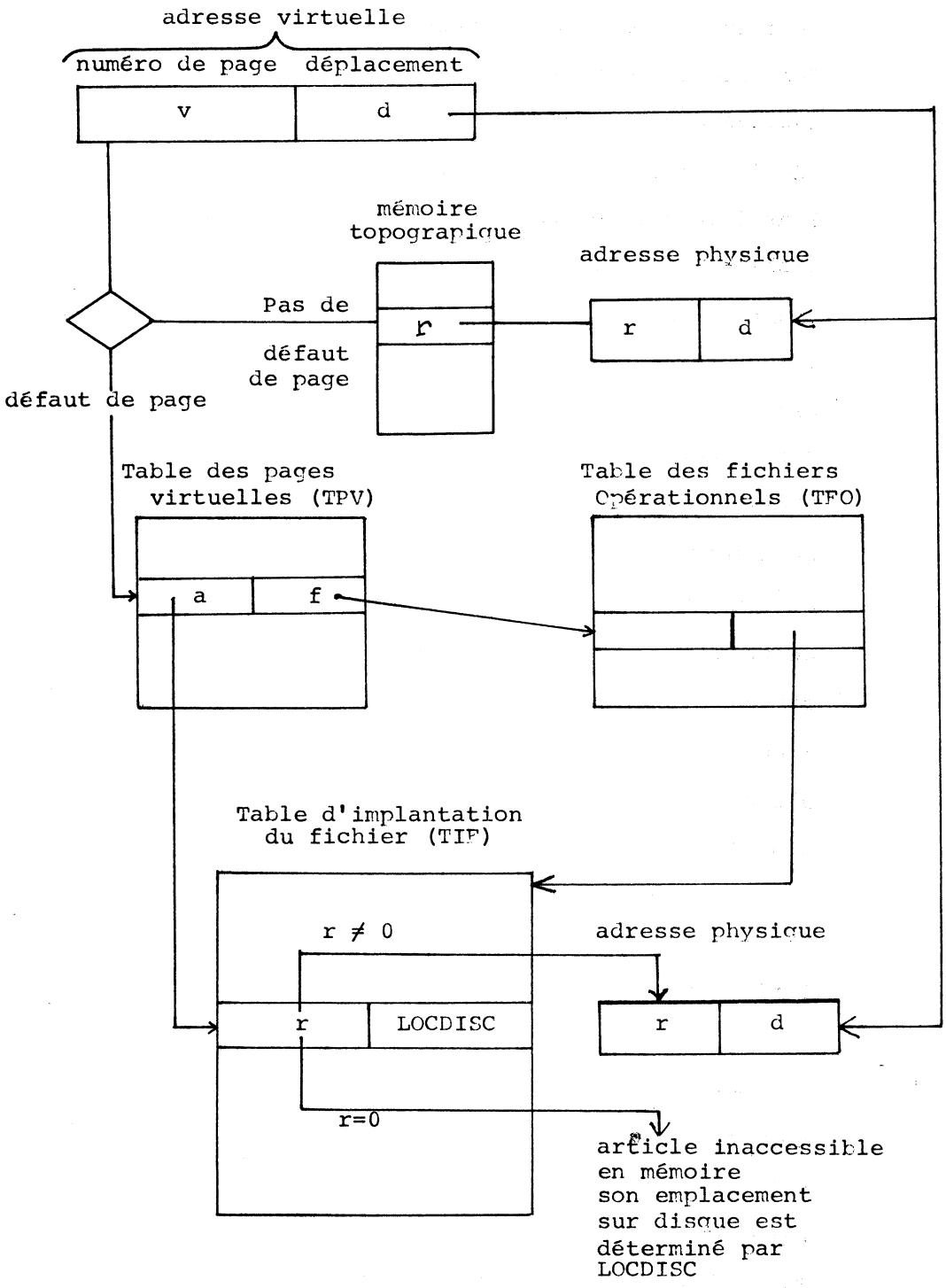
Le couplage est l'opération qui crée un couple page virtuelle-article de fichier. Il n'entraîne aucun transfert d'information entre la mémoire réelle et la mémoire secondaire. Il se fait par la primitive COUPLER (v, a, f)

v est le numéro de page virtuelle

a est le numéro de l'article dans le fichier

f est le numéro de fichier

L'entrée v de la table des pages virtuelles est mise à jour avec le numéro d'article a et le numéro de fichier f.



Mécanisme d'accès à l'information dans ESOPE

Figure 1.16

Le couplage, dans ce système, est très intéressant pour la réalisation des entrées-sorties. Tout d'abord, l'opération de couplage n'entraîne intrinsèquement aucun transfert entre la mémoire auxiliaire et la mémoire centrale : c'est seulement lors de la première référence à l'article que le transfert en mémoire centrale a lieu. Si l'article du fichier se trouve déjà en mémoire centrale, il n'y a même aucun transfert à effectuer. Les échanges entre la mémoire centrale et la mémoire auxiliaire supportant les fichiers se font en passant par un mécanisme unique de transfert de pages. Il n'y a pas à cet égard de différence entre l'accès aux articles de fichiers permanents des utilisateurs et l'accès aux pages rangées temporairement sur mémoire auxiliaire pour libérer de la place en mémoire centrale. Enfin, grâce au couplage, il est possible qu'une page de fichier située en mémoire centrale ou auxiliaire appartienne à plusieurs mémoires virtuelles. On diminue ainsi le nombre d'entrées-sorties à effectuer, et l'encombrement de la mémoire centrale par des pages identiques. Nous allons voir maintenant comment le couplage a été utilisé pour l'utilisation des périphériques classiques.

1.52 Gestion des périphériques classiques

Les périphériques autres que les disques à tête fixe et les terminaux des utilisateurs sont gérés par le Système d'Entrée-Sortie SES. Ce système d'entrée-sortie est chargé d'effectuer les transferts entre un de ces périphériques et un fichier situé sur un disque à tête fixe. Pour cela, il s'appuie sur des mécanismes généraux existant déjà dans le système : gestion de processus, gestion de fichiers, allocation de ressources. Considérons le transfert d'un fichier sur disque à tête fixe vers l'imprimante. Nous avons vu au paragraphe

précédent que l'accès à un article de fichier par un programme se fait très simplement grâce au mécanisme de couplage. Un article couplé à une page virtuelle ne réside pas de manière permanente en mémoire centrale, alors que les commandes et les données destinées à l'imprimante doivent y être résidentes et ne peuvent être soumises à la pagination. Aussi le transfert du fichier sur disque vers l'imprimante se fait par deux processus fonctionnant en parallèle. Le processus P1 transfère un article du fichier dans une page de mémoire centrale ; le processus P2 transfère le contenu de cette page sur imprimante (figure 1.17).

Le processus P1 utilise le couplage pour accéder à l'article ; il travaille en mode esclave pour que la protection de la mémoire virtuelle puisse lui être appliquée. Une page couplée n'étant pas toujours résidente en mémoire centrale, le processus P1 la transfère ensuite dans une page de sa mémoire virtuelle qui est collée à une page physique, c'est-à-dire toujours résidente en mémoire centrale. Le processus P1 ne fait ainsi explicitement aucune entrée-sortie. Seul le mécanisme de faute de page du système fait le transfert de cette page du disque à têtes fixes vers la mémoire centrale.

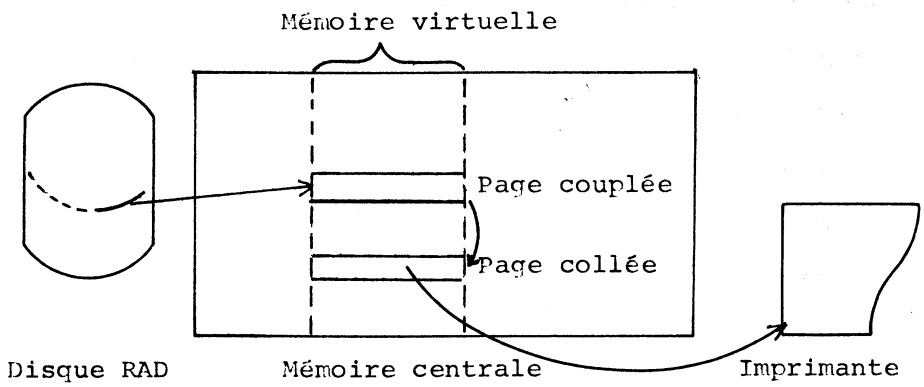
COUPLER(page virtuelle n, article a du fichier f)

RECOPIER(page virtuelle n dans page virtuelle collée p)

Le processus P2 peut alors transférer le contenu de cette page collée sur l'imprimante. Pour pouvoir utiliser les instructions privilégiées d'entrée-sortie, il s'exécute en mode maître.

La synchronisation entre le processus P2 et la sortie sur imprimante se fait par sémaphore (réf EWD). Quand le processus P2 a lancé une sortie par l'instruction SIO, il se bloque derrière un sémaphore en attendant la fin du transfert. Il est réveillé par le programme de traitement de l'interruption d'entrée-sortie.

La synchronisation entre les deux processus P1 et P2 est faite aussi par sémaphores.



Transfert d'un article de disque RAD vers imprimante

Figure 1.17

Conclusion

L'étude des systèmes que nous avons faite dans ce chapître montre de toute évidence que la gestion des entrées-sorties dans les systèmes n'est pas uniformisée. Il semble que les systèmes qui ont un superviseur d'entrée-sortie (OS/360 et MULTICS) sont bien adaptés pour gérer des ressources d'entrée-sortie à plusieurs niveaux et pour tourner sur des configurations très variées. L'étude du système CP67 met en évidence les difficultés de simulation des opérations d'entrée-sortie pour des machines virtuelles. Enfin, il est bien possible qu'ESOPE préfigure l'avenir des entrées-sorties dans les prochaines années : disparition des ordres d'entrée-sortie des programmes des utilisateurs et utilisation de la pagination pour les remplacer.

CHAPITRE 2

UNE REALISATION PARTICULIERE :

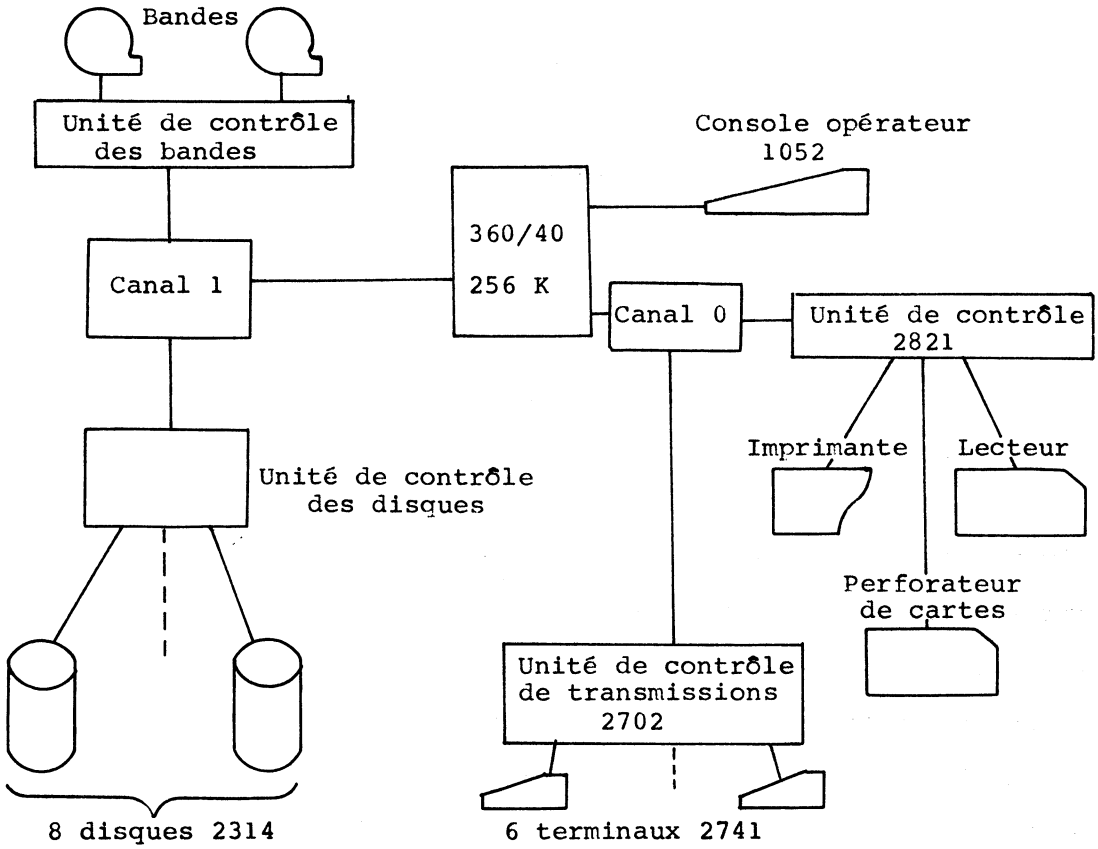
LE SUPERVISEUR D'ENTREE-SORTIE DE GMS

Nous étudions maintenant la réalisation du superviseur d'entrée-sortie de GMS, système réalisé au Centre Scientifique IBM de Grenoble. Après la présentation de GMS, la logique de réalisation du superviseur d'entrée-sortie est analysée, ainsi que les mécanismes de communication entre les programmes des utilisateurs et le superviseur d'entrée-sortie. A titre d'exemple, nous présenterons la gestion de l'imprimante sous tous ses aspects.

2.1 Présentation du système GMS

2.11 Principes de conception

Ce système a été conçu en vue de fournir à une petite communauté d'utilisateurs les ressources d'un système conversationnel. Il fonctionne sur un ordinateur IBM360 standard, sans dispositif de traduction dynamique d'adresse. Dans le cas particulier de notre réalisation, il s'agit d'un modèle 360/40, disposant de 256 K octets de mémoire principale (figure 2.1).



Configuration du Centre Scientifique IBM de Grenoble

Figure 2.1

Les unités d'entrée-sortie dans cette configuration représentent bien l'éventail des unités que l'on trouve dans la plupart des installations.

Deux solutions se présentent pour réaliser un tel système : soit concevoir entièrement un nouveau système avec tous ses composants utilitaires, soit reprendre un système conversationnel déjà existant et le rendre accessible à plusieurs utilisateurs simultanément. Nous avons préféré choisir la deuxième solution car nous disposions déjà d'un système mono-conversationnel, le système CMS, dont la logique interne était bien connue et que nous pouvions modifier

facilement (1.21). Pour fonctionner, il a besoin de toutes les ressources de l'unité centrale, exception faite de l'horloge et des instructions de protection mémoire. Pour rendre CMS accessible à plusieurs utilisateurs, deux approches sont possibles. On peut modifier le système CMS pour qu'il gère lui-même le partage des ressources de l'ordinateur. Cette solution est difficile à réaliser car CMS n'a pas de mécanisme formel d'allocation de ressources.

La deuxième solution que nous avons retenue consiste à laisser CMS contrôler un seul utilisateur et à ajouter entre le système CMS et l'utilisateur un autre niveau : nous y plaçons le système GMS que nous appelerons un hyperviseur (figure 2.2).

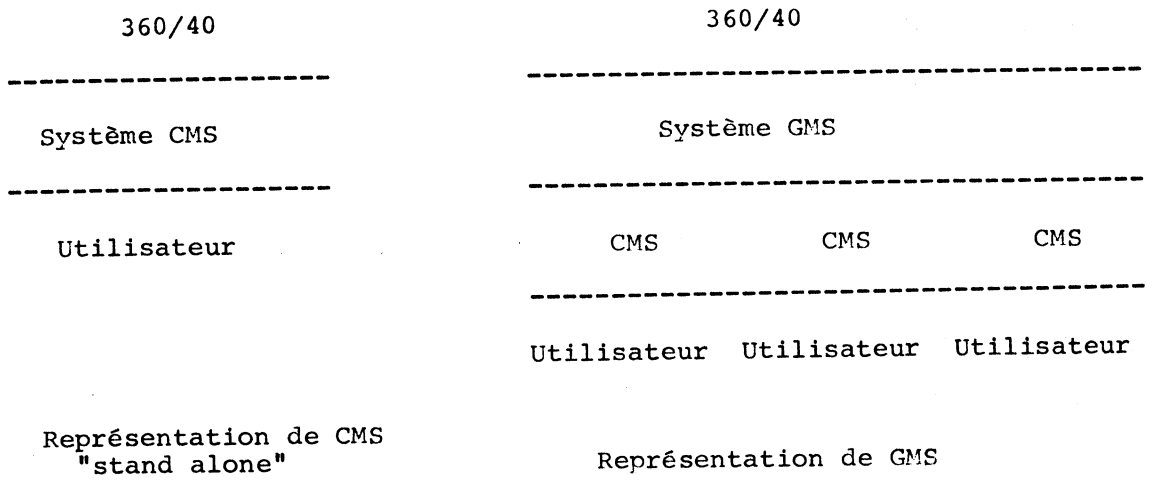


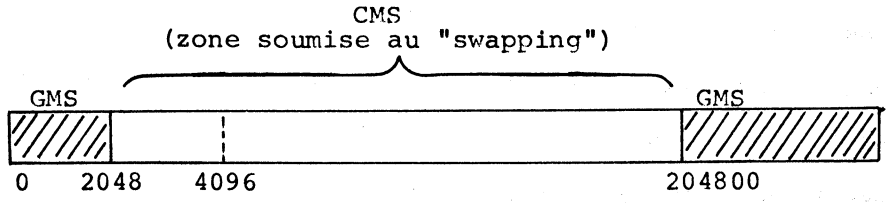
Figure 2.2

Sur cette figure, chaque couche représente un niveau. Chaque niveau peut utiliser les ressources du niveau adjacent supérieur. Les ressources du système CMS (compilateurs, éditeurs, ...) sont ainsi accessibles à l'utilisateur, tandis que les ressources de l'ordinateur (unité centrale, unités d'entrée-sortie) sont accessibles au système CMS "stand-alone".

GMS a la charge de générer pour CMS toutes les ressources dont celui-ci disposait quand il fonctionnait en "stand-alone". Il partage les ressources de l'ordinateur entre plusieurs copies de CMS. Une copie active peut être créée par tout utilisateur présent à un terminal connecté.

2.12 Allocation de la mémoire centrale

Cette génération de ressources par GMS se rapproche de la génération des machines virtuelles par le système CP67. Toutefois des différences fondamentales apparaissent entre les deux systèmes (réf BH). Dans une machine virtuelle, il est toujours possible de faire une référence et de modifier tout élément de la mémoire virtuelle. Par exemple, une référence virtuelle à l'adresse 0 est transformée par le mécanisme de pagination en une référence réelle à l'adresse a. La mémoire basse de l'ordinateur (qui correspond aux plus petites adresses) reste sous le contrôle exclusif de CP67. Dans GMS, ceci est impossible. Aussi le système CMS fonctionnant sous GMS a été modifié pour qu'il ne fasse jamais référence à la partie basse de la mémoire réelle (adresses 0-2048). L'information qui se trouvait entre ces deux adresses a été translatée dans les adresses 2048-4096 (figure 2.3).



Allocation de la mémoire centrale

Figure 2.3

La mémoire haute est réservée à GMS pour y implanter le noyau et contenir les zones dynamiques acquises par GMS en cours de session.

Le reste de la mémoire centrale est allouée à CMS. C'est l'espace adressable des utilisateurs. Comme il n'y a pas de translation dynamique d'adresse ni de protection en lecture, on ne peut faire de la multiprogrammation : en effet, dans CMS "stand-alone", toute la mémoire centrale est accessible et modifiable par l'utilisateur. Aussi, dans GMS, la mémoire centrale est allouée en un bloc unique aux utilisateurs. Code et données de chacun sont transférés à tour de rôle entre mémoire centrale et mémoire secondaire. Pour celà, deux cylindres sont reservés par utilisateur sur un disque 2314, et on pratique le "swapping" global de toute la zone utilisateur pour partager la mémoire.

2.13 Allocation du temps d'unité centrale

L'unité centrale est une ressource unique à partager entre tous les utilisateurs. La technique employée est celle du temps partagé avec préemption. Un utilisateur qui reçoit le contrôle de l'unité centrale se voit attribuer une tranche de temps : quand celle-ci est épuisée, le contrôle du processeur central est donné à un autre utilisateur. L'utilisation de la préemption permet à un utilisateur émettant des commandes interactives d'interrompre l'utilisateur ayant le contrôle de l'unité centrale, avant que la tranche de temps de ce dernier ne soit épuisée. L'algorithme que nous avons conçu tient compte de la préemption, tout en garantissant aux utilisateurs non interactifs un pourcentage minimum d'utilisation de l'unité centrale (réf BP, LF).

2.14 Allocation des ressources d'entrée-sortie

GMS doit générer, pour chaque copie de CMS, les unités d'entrée-sortie dont celles-ci ont besoin. La configuration minimum sur laquelle tourne CMS (figure 2.4) est la suivante :

- une console opérateur par laquelle l'utilisateur converse avec le système.
- un disque système sur lequel sont conservés les fichiers dont le système a besoin en cours de session.
- un disque sur lequel l'utilisateur conserve ses fichiers personnels.
- un lecteur-perforateur de cartes.
- une imprimante.
- deux unités de bande.

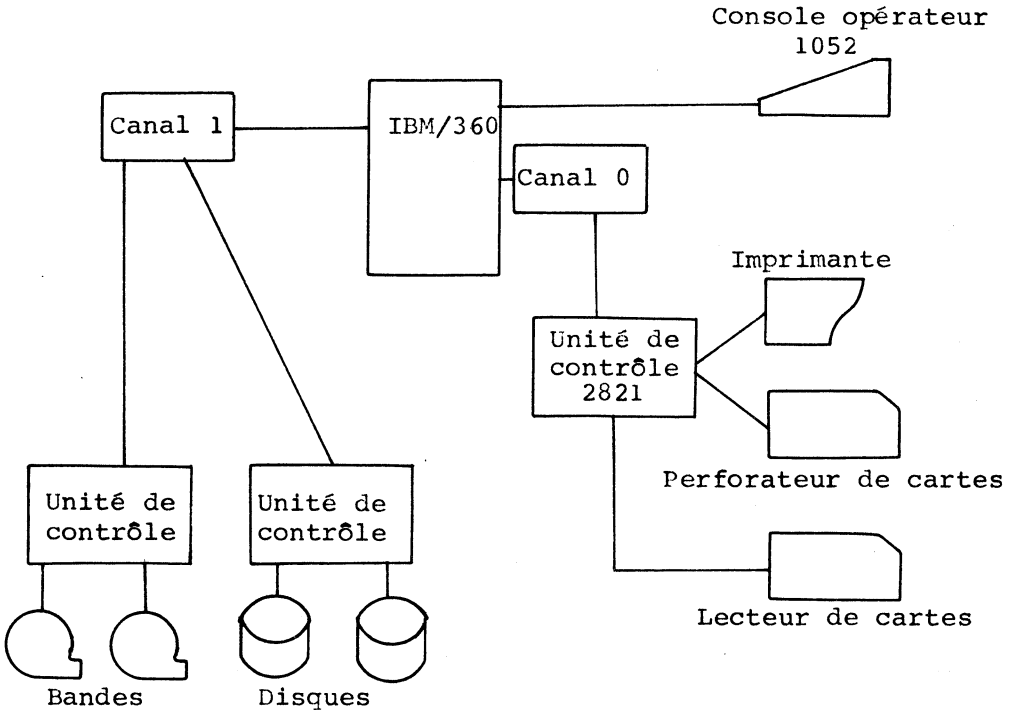


Figure 2.4 Configuration nécessaire à CMS

Divers types de génération sont possibles pour les unités virtuelles d'entrée-sortie (réf LF).

- On peut associer à une unité virtuelle une unité réelle de même type. C'est la solution adoptée pour les disques et les bandes.
- On peut associer à une unité virtuelle une unité réelle d'un autre type : la console opérateur est ainsi simulée sur un terminal 2741.
- Une unité virtuelle peut être simulée par une unité logique sans support physique. C'est le cas de l'imprimante et du lecteur-perforateur de cartes, qui sont en fait des fichiers de "spooling" gérés par GMS.

Quelque soit le type de génération adopté, il faut un interface qui établisse la communication entre le virtuel et le réel. C'est cet interface qui fait appel au superviseur d'entrée-sortie de GMS pour faire les opérations d'entrée-sortie réelles.

2.15 Classification des tâches dans GMS

Les programmes qui utilisent l'unité centrale ne disposent pas tous des mêmes ressources, ni de la même priorité. Ils sont divisés en trois classes :

1- Les programmes de traitement d'interruption.

L'exécution de ces programmes a lieu dès que les interruptions correspondantes se produisent, car toute interruption non traitée est perdue. De plus, une interruption peut modifier l'ordonnancement des programmes des deux autres classes. Certaines interruptions peuvent être masquées sélectivement par le système (par exemple les interruptions d'entrée-sortie). On peut ainsi modifier la priorité des programmes de cette classe.

2- Les tâches système.

A un instant donné, les tâches système représentent les programmes du noyau de GMS qui demandent ou qui possèdent l'allocation de l'unité centrale. Elles ne sont pas soumises au partage de temps, car on admet que le système est bien écrit et que ces tâches n'ont besoin de l'unité centrale que pendant un temps limité. Elles sont prioritaires par rapport aux programmes de la classe 3 car elles gèrent les ressources réelles de la machine (par exemple unités d'entrée-sortie). Toutes les tâches système, à l'exception de celle qui a le contrôle de l'unité centrale, sont connues dans le système par leur descripteur (figure 2.5). Ce descripteur est composé de deux mots dont la structure est la même pour toutes les tâches système, et d'une extension dont la longueur et le contenu varient suivant les tâches.

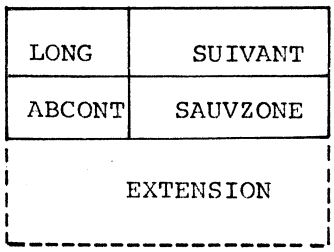


Figure 2.5 Descripteur de tâche système

LONG : longueur du descripteur

SUIVANT : adresse du descripteur de la prochaine tâche à activer

ABCONT : option indiquant si la place occupée par le descripteur doit être rendu à la mémoire libre du système quand la tâche est activée.

SAUVZONE : adresse de la zone contenant la valeur des registres à charger avant réactivation de la tâche.

L'adresse du point d'entrée du programme à activer se trouve, par convention, dans le registre 14.

Les tâches de cette classe se trouvent dans une file d'attente de type FIFO (first in - first out : premier arrivé, premier servi). L'adresse de la tête de cette file est dans le pointeur GMSFSTK dont l'emplacement est fixé à la génération du système. Leur activation est faite par le contrôleur qui gère aussi les programmes de la troisième classe. Le contrôleur est appelé après le traitement de tout évènement susceptible d'avoir modifié les priorités des tâches.

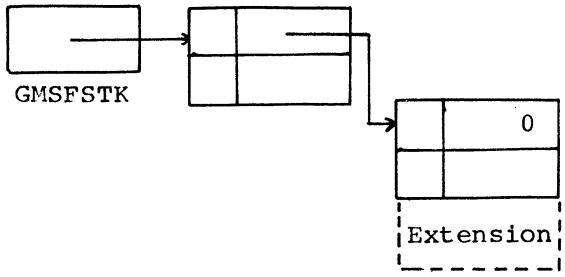


Figure 2.6 File d'attente des tâches système

3-Les tâches utilisateur.

Une telle tâche est créée par un utilisateur à son terminal lorsqu'il demande la création d'une copie du système CMS dans son espace adressable. L'activation d'une tâche utilisateur est l'exécution d'un programme de cette copie de CMS ou d'un programme propre à l'utilisateur. C'est à ces tâches-là que s'appliquent la technique du temps partagé et le principe de préemption. Elles sont connues dans le système par leur descripteur (figure 2.7). Il contient en particulier :

- le nom de l'utilisateur
- pour chaque unité d'entrée-sortie virtuelle, une description du support réel correspondant.

- une zone de sauvegarde du contexte de la tâche, c'est-à-dire des informations permettant de réactiver la tâche dans l'état où elle était lorsqu'elle a été désactivée : mot d'état programme + registres.
- plusieurs indicateurs permettant de connaître l'état de la tâche dans le système.
- un mot contenant le temps non utilisé d'une tranche allouée à la tâche.
- un pointeur vers une autre tâche activable.

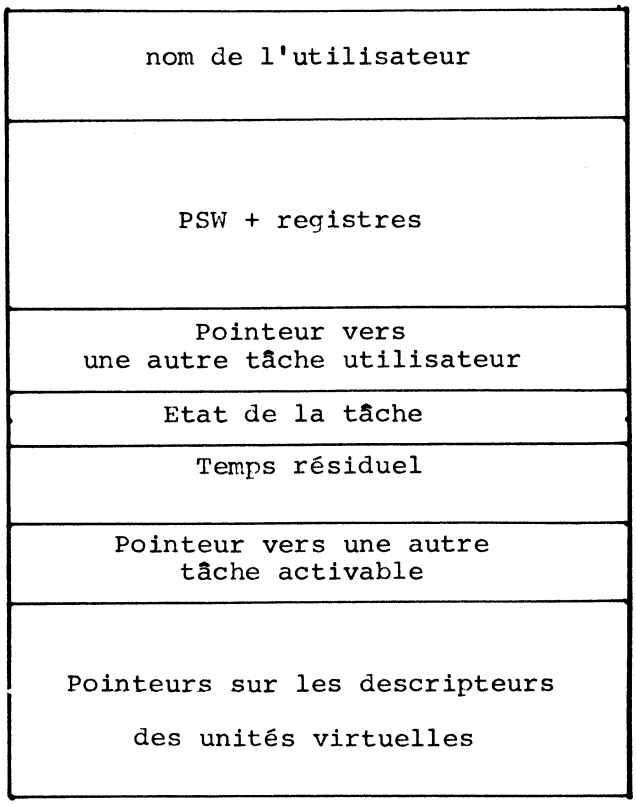


Figure 2.7 Principaux éléments d'un descripteur de tâche utilisateur

2.16 Environnements

Un environnement est un ensemble de ressources fournies à un programme qui s'exécute. Le terme "ressources" doit s'entendre ici dans un sens très général. Ce peut être l'unité centrale elle-même, dans un mode déterminé (maître ou esclave), aussi bien qu'un système entier ou une partie de système.

Une tâche utilisateur peut se trouver dans l'environnemnt GMS ou dans l'environnement CMS (figure 2.8). Elle est dans l'environnement GMS quand l'utilisateur vient de se connecter au système, ou quand il est sorti de l'environnement CMS en appuyant sur la touche "Attention" du terminal : c'est une convention du système. L'utilisateur a accès aux commandes de GMS, mais il ne peut faire exécuter ses programmes.

Elle est dans l'environnement CMS dès que l'utilisateur a activé sa copie du système CMS, soit par "IPL" de CMS, soit par simple sortie de l'environnement GMS. Quand un programme s'exécute dans cet environnement, l'ordinateur travaille en mode esclave, et la protection mémoire qui lui est appliquée lui interdit la modification des zones réservées à GMS.

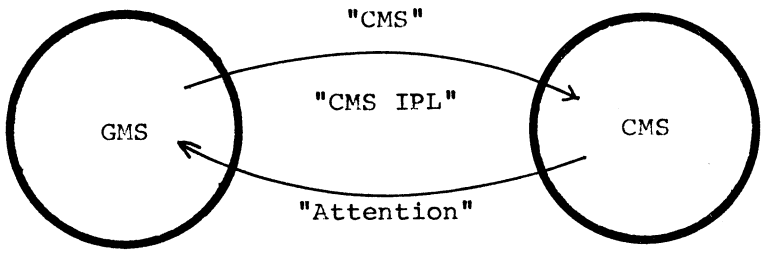


Figure 2.8 Environnements des tâches utilisateurs

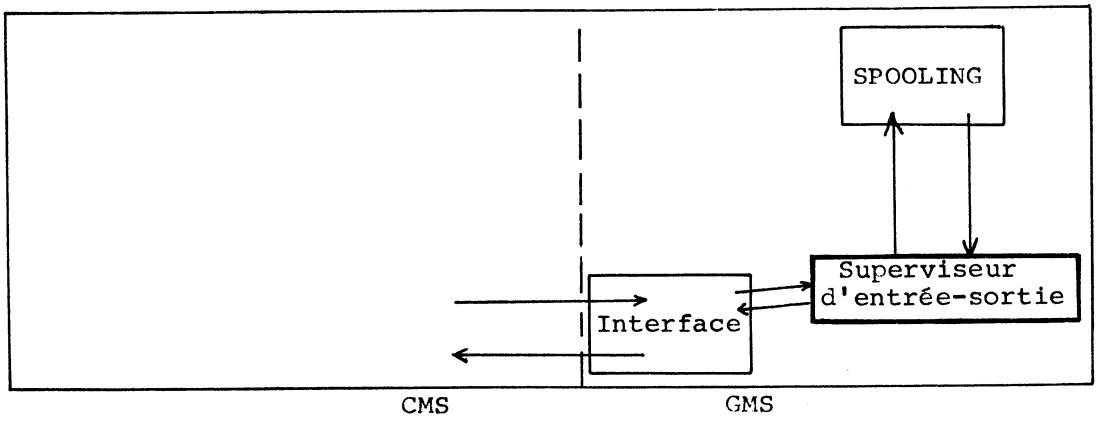
C'est aux procédures appelées par le superviseur d'entrée-sortie qu'il appartient de détecter ces changements d'environnements.

2.2 Logique du superviseur d'entrée-sortie

Toutes les opérations d'entrée-sortie de GMS se font par appel à un programme spécialisé unique, le superviseur d'entrée-sortie (réf JB). La gestion globale de toutes les unités d'entrée-sortie et des chemins qui y mènent (canaux, unités de contrôle) lui est réservée : le partage d'unités ou de canaux par des programmes indépendants ne crée donc pas de risque de blocage. Il peut optimiser la gestion des unités puisqu'il connaît en permanence les demandes d'entrée-sortie sur les unités : il exploite le parallélisme de fonctionnement des unités.

Deux classes d'opérations d'entrée-sortie sont à distinguer dans GMS (figure 2.9) :

- les opérations d'entrée-sortie résultant de la simulation des entrées-sorties virtuelles de CMS sur les disques, les bandes et les terminaux. CMS transmet ces demandes à GMS par une instruction SIO qui crée une interruption programme. Un interface de GMS prend le contrôle, vérifie la validité de ces demandes, les traduit en opérations d'entrée-sortie, et appelle le superviseur d'entrée-sortie.
- lles opérations d'entrée-sortie propres à GMS ("spooling", conversation système-opérateur par le 1052). Les modules qui effectuent ces fonctions appellent directement le superviseur d'entrée-sortie.



Utilisation du superviseur d'entrée-sortie
dans GMS

Figure 2.9

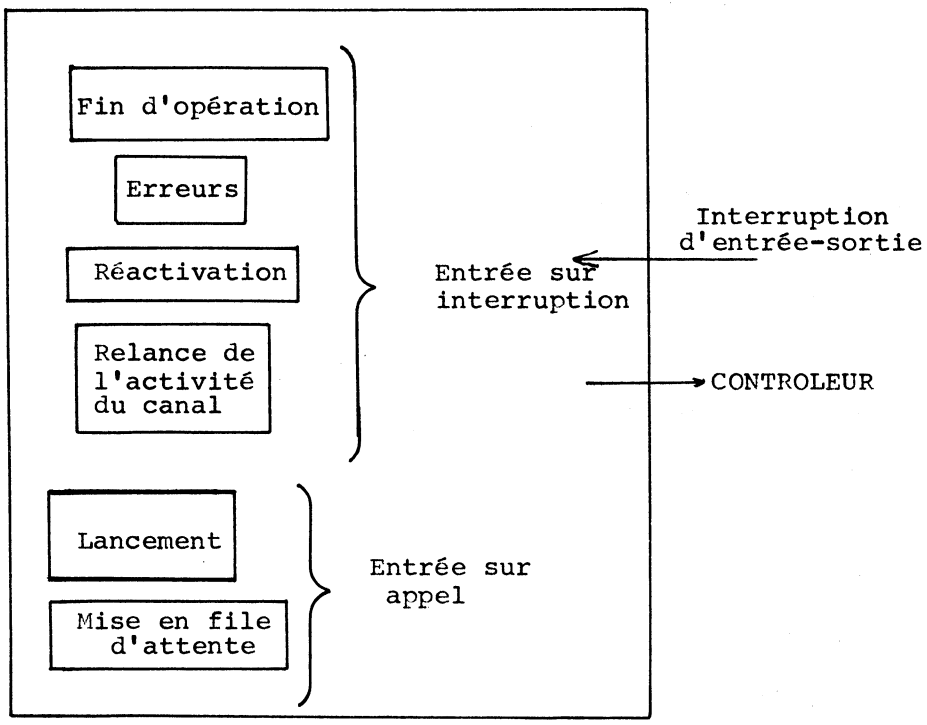
Le superviseur d'entrée-sortie doit pouvoir se conformer aux particularités de toutes ces demandes d'entrée-sortie. Il résoud deux problèmes principaux :

-Synchronisation des entrées-sorties avec des programmes du système et des utilisateurs via l'interface CMS-GMS.

-Ingérence des utilisateurs dans le traitement des entrées-sorties.

Une demande d'opération d'entrée-sortie est soumise au superviseur d'entrée-sortie sous la forme d'une tâche d'entrée-sortie. Le superviseur d'entrée-sortie la met dans la file d'attente rattachée à l'unité impliquée dans l'opération et elle y reste tant que l'opération n'est pas achevée ou suspendue. L'opération est initialisée dès que le chemin menant à l'unité est libre. En plus de cette prise en charge des opérations d'entrée-sortie, le superviseur d'entrée-sortie doit traiter les interruptions d'entrée-sortie. A ce niveau, il faut distinguer deux rôles du superviseur d'entrée-sortie :

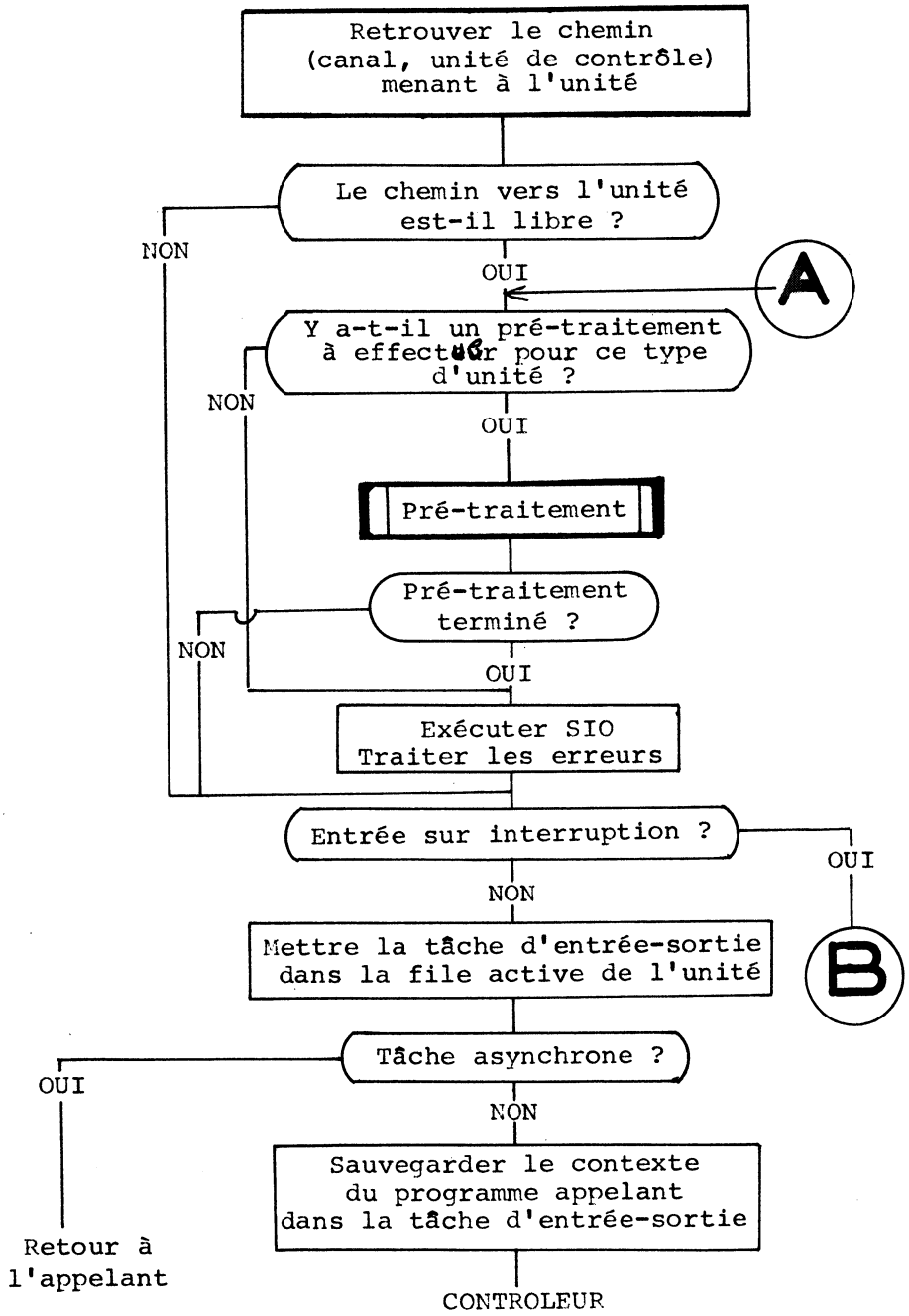
- Analyse de l'interruption, et synchronisation éventuelle avec le programme émetteur de l'entrée-sortie correspondante : le superviseur doit déterminer l'état d'achèvement de l'entrée-sortie et faire réveiller les tâches qui attendaient cet achèvement.
- Relance de l'activité du canal : une interruption marquant généralement la libération d'une unité ou d'un canal, il faut essayer de les occuper à nouveau immédiatement. Les tâches d'entrée-sortie rattachées aux unités permettent de relancer des opérations sur des unités disponibles. C'est ici qu'intervient l'optimisation d'emploi des unités.



Organisation générale
du superviseur d'entrée-sortie

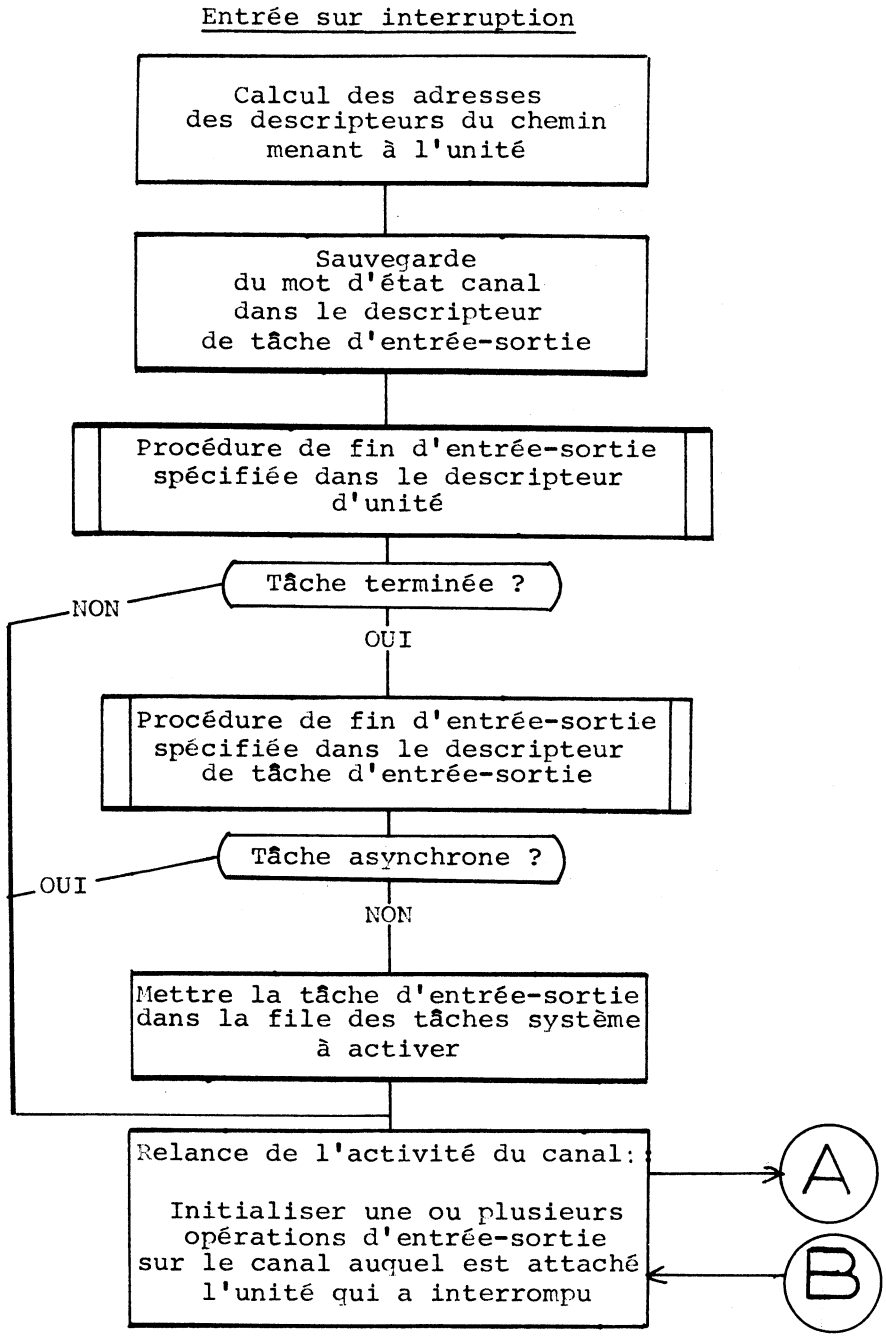
Figure 2.10

Entrée sur appel



Traitement d'une tâche d'entrée-sortie
par le superviseur d'entrée-sortie

Figure 2.11 (1/2)



CONTROLEUR

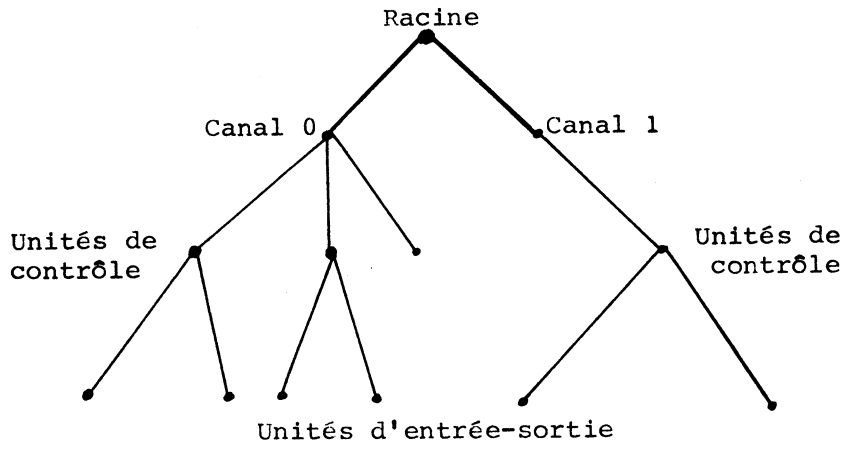
Traitement d'une tâche d'entrée-sortie
par le superviseur d'entrée-sortie

Figure 2.11 (2/2)

2.21 Représentation de la configuration dans le système

La configuration de l'ordinateur doit être représentée en mémoire de manière à faire apparaître les caractéristiques de chaque unité ou de chaque canal. Nous avons adopté une représentation en arbre pour indiquer la dépendance d'une unité vis-à-vis d'un canal et d'une unité de contrôle. Cette représentation permet de retrouver, à partir d'une adresse d'unité "abc" le chemin qui mène de la racine de l'arbre à l'unité.

Nous appelons ici "module E/S" un quelconque des trois éléments suivants : canal, unité de contrôle, unité d'entrée-sortie.

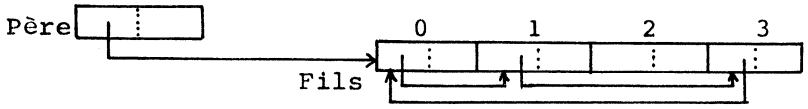


Représentation de la configuration en arbre

Figure 2.12

Les modules E/S desservis par un module E/S commun sont "frères", et le module commun par lequel ils sont desservis est leur "père". Par exemple, les terminaux 2741 sont "frères" et l'unité de contrôle 2702 est leur "père". Les modules frères se trouvent dans une liste dont les entrées occupent des emplacements de mémoire consécutifs.

L'index dans cette liste est donnée par un des éléments de l'adresse physique de l'unité :
"abc" étant l'adresse physique de l'unité,
"a" est l'index dans la liste des canaux,
"b" est l'index dans la liste des unités de contrôle,
et "c" est l'index dans la liste des unités. Dans le descripteur du module "père", on trouve un pointeur vers le descripteur de son fils d'index 0, et un pointeur vers un autre "père".

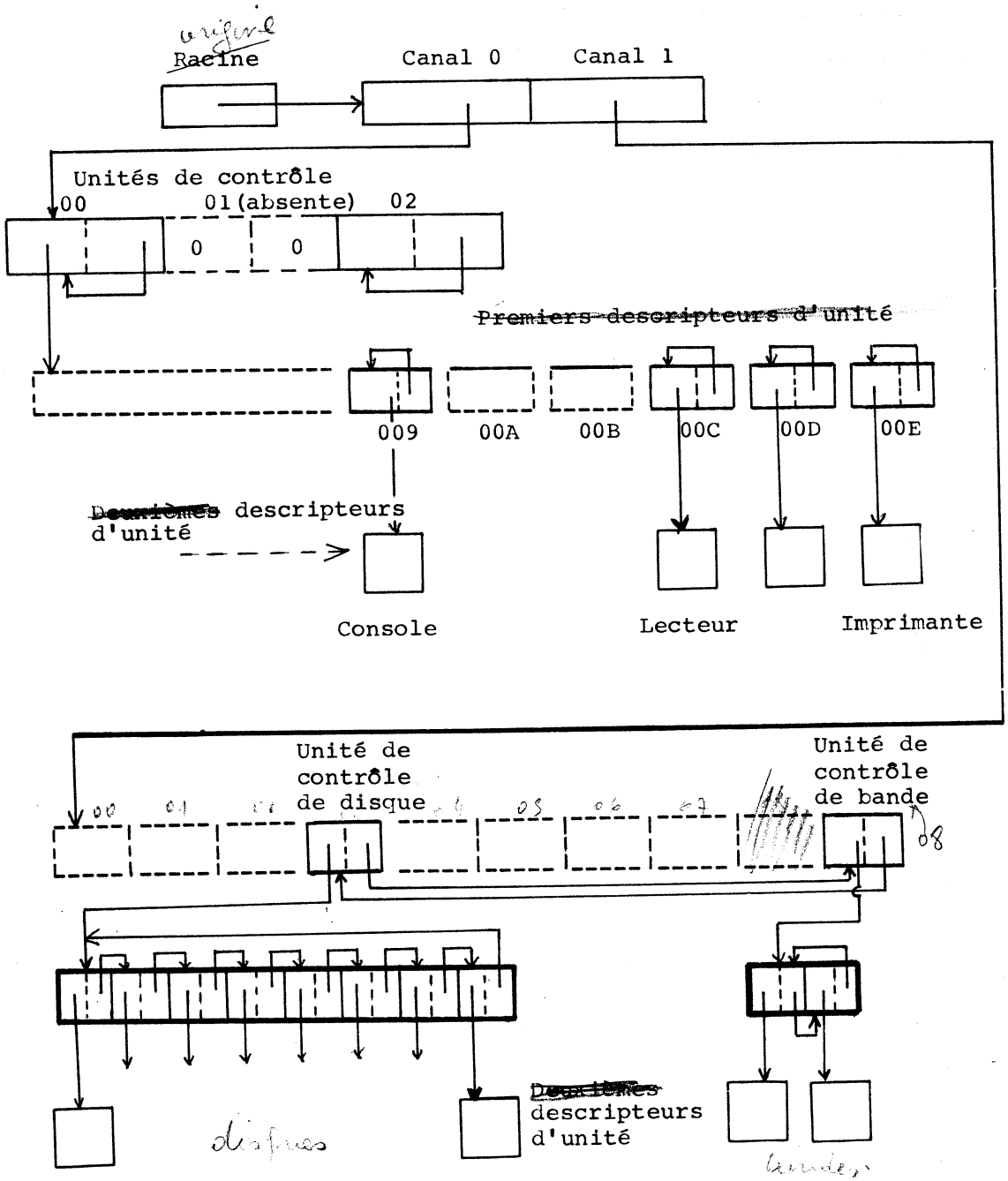


Relations Père - Fils

Figure 2.13

A étant l'adresse en mémoire centrale du fils d'index 0, et T la taille d'un descripteur, l'adresse du descripteur de son frère d'index "i" est : A+T.i

Dans ces listes séquentielles, dont tous les éléments peuvent d'ailleurs ne pas correspondre à un module E/S existant dans la configuration, on trouve des sous-listes Lk : Lk est une liste chaînée d'éléments ne pouvant utiliser simultanément le module "père". Par exemple, un canal sélecteur ne peut desservir qu'une unité de contrôle à un instant donné : les unités de contrôle ayant un canal sélecteur comme "père" sont donc chaînées en une sous-liste unique. Par contre, l'unité de contrôle 2821 pouvant desservir simultanément lecteur de cartes, perforateur et imprimante, chacune de ces unités est décrite dans une sous-liste à un seul élément. Remarque : cette représentation n'est pas applicable aux unités accessibles par plusieurs chemins, par exemple par deux canaux.



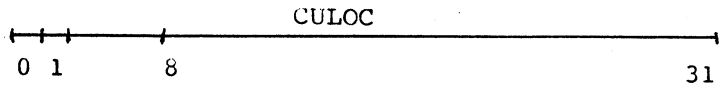
Tables de représentation de la configuration

Figure 2.14

Quand une interruption vient de se produire, marquant ainsi la disponibilité d'un module E/S, une telle organisation permet de trouver une unité sur laquelle le lancement d'une opération dépendait de cette libération. Elle présente en outre l'avantage d'une grande rapidité de consultation, par rapport à une simple organisation en listes séquentielles, où toutes les unités figureraient dans une seule liste. Le risque que l'unité dont on connaît l'adresse physique ne soit pas dans les tables existe, par exemple si une unité n'existait pas dans la configuration à la génération du système. Pour le supprimer, il suffit de vérifier, chaque fois qu'un descripteur est sélecté, que les fils de ce descripteur existent.

Description des tables

Le descripteur d'un canal est formé d'un mot.

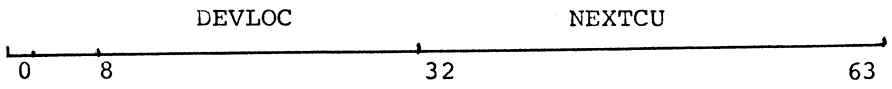


Le bit 0 indique l'occupation du canal : il est mis à 1 après le lancement d'une opération d'entrée-sortie sur ce canal, s'il est sélecteur. Il est remis à 0 quand se produit l'interruption "Fin sur canal".

Le bit 1 est utilisé de manière temporaire dans la partie "relance activité canal" du superviseur d'entrée-sortie.

Les bits 8-31 représentent l'adresse en mémoire centrale de la liste des descripteurs d'unités de contrôle rattachées à ce canal.

Le descripteur d'une unité de contrôle occupe un double mot.



Le bit 0 indique l'état d'occupation de l'unité de contrôle.

Les bits 8-31 contiennent l'adresse du descripteur de l'unité d'index 0 rattachée à cette unité de contrôle.

Les bits 32-63 contiennent l'adresse du prochain élément de la liste des unités de contrôle rattachées au canal.

Une unité d'entrée-sortie est décrite par deux descripteurs. Ceci est motivé par le fait que, dans la suite des descripteurs, tous ne correspondent pas à des unités existant réellement dans le système. Il est inutile de réserver une trop grande place pour ces descripteurs. Le premier descripteur d'une unité est un double mot : le premier mot est un pointeur vers le deuxième descripteur, Le deuxième mot est un pointeur vers le prochain élément de la liste des premiers descripteurs (figure 2.15).

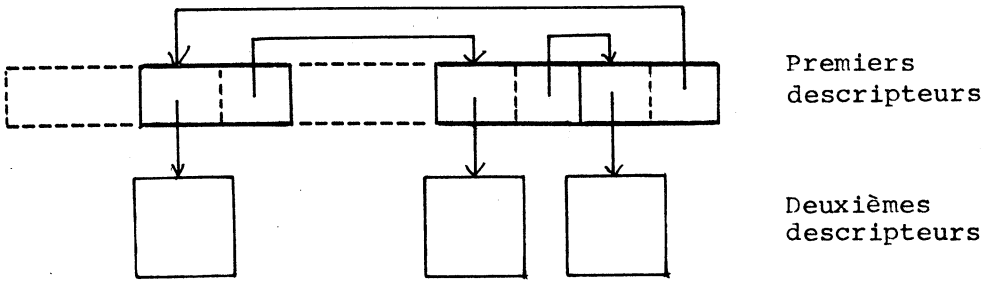
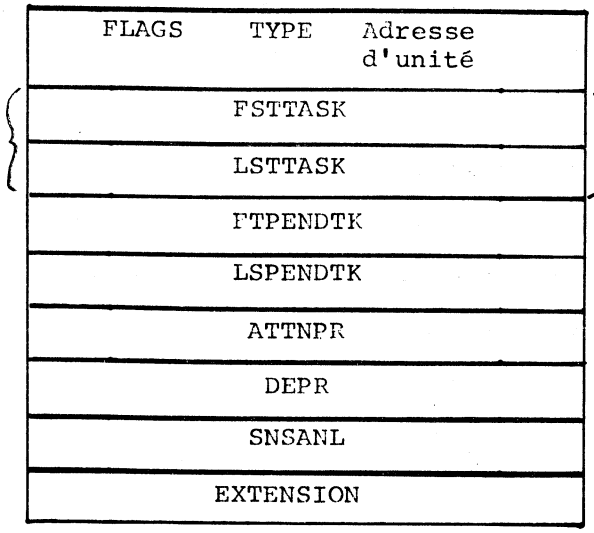


Figure 2.15 Descripteurs d'unité

Le deuxième descripteur contient tous les renseignements utiles concernant l'unité (figure 2.16). Sa structure est la même pour toutes les unités. Les informations valables seulement pour un type d'unité sont décrites dans une extension dont l'adresse se trouve dans le deuxième descripteur. Désormais, sauf précision contraire, le terme "descripteur d'unité" désignera toujours le deuxième descripteur d'unité.



FLAGS — état de l'unité
- Bit 0 BUSY 1==> unité occupée
- Bit 1 DVNR 1==> unité non prête
- Bit 2 SENSE 1==> analyse en cours sur l'unité
- Bit 3 BURS 1==> l'unité utilise les services du canal
- Bit 4 HPND 1==> un arrêt d'opération est en cours sur l'unité
- Bit 5 SNSP 1==> opération d'analyse en attente

TYPE — type de l'unité :
- disques
- bandes
- lecteur-perforateur, imprimante, console 1052
- terminaux 2741

FSTTASK — Pointeur sur la première tâche active sur l'unité

LSTTASK — Pointeur sur la dernière tâche active sur l'unité

FTPENDTK — Pointeur sur la première tâche suspendue

LSPENDTK — Pointeur sur la dernière tâche suspendue

ATTNPR — Adresse de procédure traitant les interruptions asynchrones

DEPR — Adresse de procédure traitant les interruptions synchrones indiquant un fin normale d'entrée-sortie

SNSANL — Adresse de procédure traitant les informations d'analyse

EXTENSION — Adresse d'un bloc de contrôle contenant des informations propres à l'unité

Figure 2.16 Deuxième descripteur d'unité

2.22 Activation du superviseur d'entrée-sortie par appel

2.221 Paramètres

Un programme qui veut effectuer une entrée-sortie appelle le superviseur d'entrée-sortie en lui indiquant :

1- sur quelle unité lancer l'entrée-sortie.

2- la nature de l'opération d'entrée-sortie (programme canal, options).

En sens inverse, le processus émetteur peut désirer recevoir des informations sur la manière dont s'est déroulée l'entrée-sortie.

Pour indiquer l'unité d'entrée-sortie, il suffit de fournir en paramètre de l'appel l'adresse du deuxième descripteur de l'unité ; on pourrait fournir l'adresse physique de l'unité, puisqu'il y a correspondance bi-univoque entre les deux. La première solution est toutefois préférable car l'identification d'une unité dans les tables du système figure généralement sous cette forme-là. L'adresse physique n'est indispensable que dans le superviseur d'entrée-sortie (pour SIO et les traitements d'interruptions). Ce dernier est donc le seul module du système à retrouver dans l'arbre des descripteurs le chemin menant à une unité dont on connaît l'adresse physique.

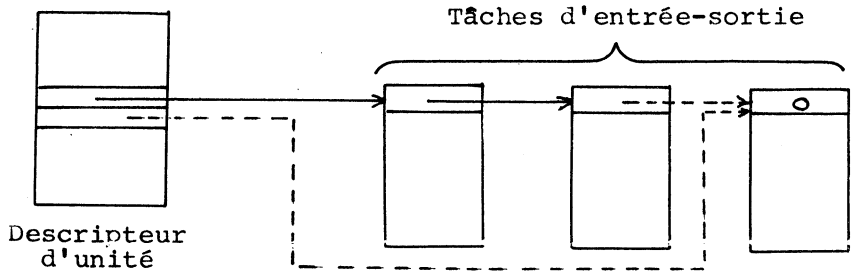
Le superviseur d'entrée-sortie étant toujours appelé par un programme du système GMS, on se fie à ces programmes pour fournir une adresse correcte de descripteur d'unité. On peut ainsi supprimer à ce niveau la vérification de la validité des paramètres. Dans un système où les utilisateurs peuvent appeler le superviseur d'entrée-sortie, tel que OS/360, les utilisateurs ne peuvent avoir accès aux descripteurs d'unités UCB. Ils précisent les unités sur lesquelles ils veulent travailler par l'intermédiaire du bloc de

contrôle de données, dont ils mettent l'adresse dans le bloc d'entrée-sortie IOB. C'est seulement à l'appel de la procédure OPEN que le système fait la liaison entre le DCB et les UCB. Les descripteurs d'unité sont ainsi protégés en permanence contre les erreurs des utilisateurs.

Une tâche d'entrée-sortie est une opération d'entrée-sortie demandée par un programme de GMS. Elle est représentée dans le système par un descripteur : il contient les informations relatives à la nature de l'opération, et le superviseur d'entrée-sortie y met les informations qu'il veut retourner au programme appelant.

2.222 Descripteur de tâche d'entrée-sortie

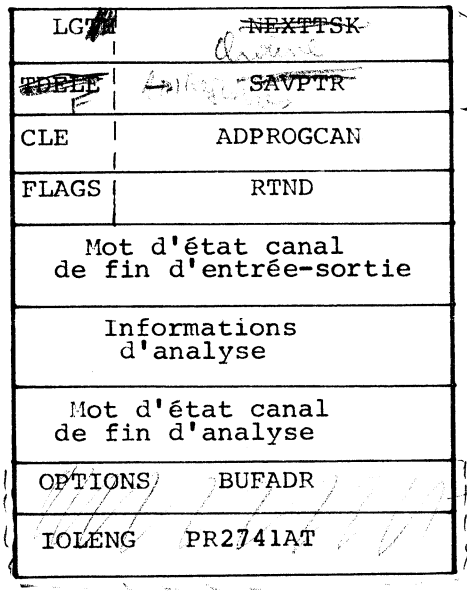
Quand elle est soumise au superviseur d'entrée-sortie, ce dernier l'ajoute à la file active des tâches d'entrée-sortie chaînée à l'unité sur laquelle elles doivent s'exécuter (figure 2.17). Le premier mot du descripteur de la tâche est utilisé pour ce chaînage.



En pointillé : chaînage effectué par le superviseur d'entrée-sortie

File active des tâches d'entrée-sortie

Figure 2.17



LGT Longueur du descripteur

NEXTTSK Pointeur de chaînage dans les files d'attente

TDELE Ce bit vaut 1 si la place occupée par le descripteur doit être restituée lorsque le contrôleur réactive la tâche système ayant créé cette tâche d'entrée-sortie

SAVPTR Adresse de la zone de sauvegarde des registres

CLE Clé de protection appliquée à l'entrée-sortie
La valeur de cette clé est contrôlée par le superviseur d'entrée-sortie.

ADPRGCAN Adresse du programme canal

FLAGS Cet octet contient les options de la tâche d'entrée-sortie, ainsi que des variables utilisées de façon interne dans le superviseur d'entrée-sortie
Bit 0 SCR 1 === Tâche auto-correctrice
Bit 1 DELETE 1 === Descripteur à libérer par RTND
Bit 2 PRSIO 1 === Prétraitement achevé
Bit 3 RCOMPL 1 === Tâche terminée
Bit 4 CSWSTO 1 === Mot d'état canal rangé sur SIO
Bit 5 WAITIO 1 === Tâche synchrone

OPTIONS Options pour l'édition des lignes

BUFADR Adresse de la ligne à écrire ou à lire sur terminal

IOLENG Longueur de cette ligne
PR2741AT Adresse de la procédure traitant "Attention" sur lecture ou écriture

Figure 2.18 Descripteur de tâche d'entrée-sortie

2.2221 Tâches synchrones et asynchrones

Le programme qui fait appel au superviseur d'entrée-sortie peut demander que l'entrée-sortie se déroule de manière synchrone ou asynchrone. Elle se déroule de manière synchrone si le programme appelant est mis en sommeil tant que l'entrée-sortie n'est pas entièrement terminée. Elle se déroule de manière asynchrone si le programme appelant reprend le contrôle dès que le superviseur d'entrée-sortie a pris acte de la demande d'entrée-sortie.

Une entrée-sortie nécessite toujours des ressources du système, statiques ou acquises dynamiquement : par exemple, la zone de mémoire contenant les données à sortir sur imprimante est une ressource du système. Ces ressources particulières à l'opération d'entrée-sortie ne doivent pas être partagées. Dans le cas d'une entrée-sortie synchrone, cette protection est facilement faite. Il suffit que la ressource soit réservée pour le programme appelant avant qu'il n'appelle le superviseur d'entrée-sortie, et qu'il la libère quand ce dernier lui rend le contrôle, à la fin de l'entrée-sortie (figure 2.19).

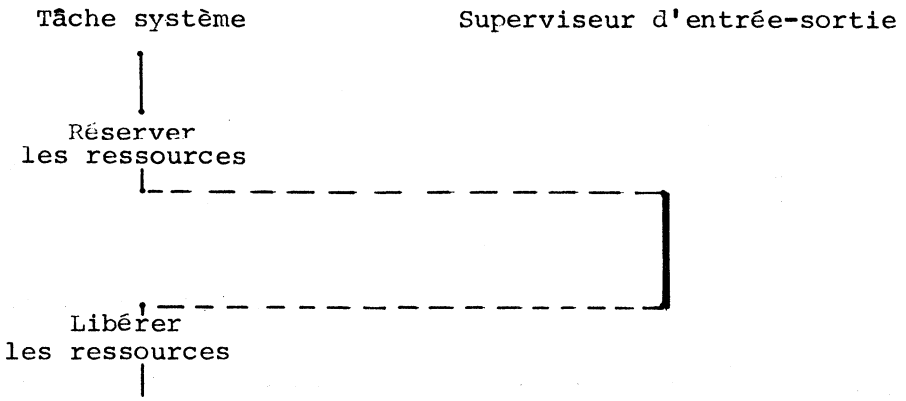


Figure 2.19 Utilisation des ressources pour une entrée-sortie synchrone

Dans le cas d'une entrée-sortie asynchrone, il faut disposer d'un exemplaire des ressources non partageables dont elle a besoin (par exemple, mémoire tampon pour sortie sur terminal). Hélas, ces ressources ne sont pas inépuisables et il peut arriver qu'un programme émettant une demande d'entrée-sortie asynchrone doive attendre la libération d'une ressource.

Les entrées-sorties asynchrones sont employées généralement dans deux situations :

- soit pour sortir des données sur un support telles qu'elles ne puissent être réexploitées sans intervention d'un opérateur : opérations sur imprimante, lecteur de cartes, terminal, ... Il n'y a pas à protéger ces données sur l'unité. D'autre part, compte-tenu de la lenteur de ces unités relativement à la vitesse de l'unité centrale, l'asynchronisme permet de ne pas bloquer une tâche pendant un temps trop long : ceci est particulièrement important s'elle a, par ailleurs, mobilisé des ressources non partageables attendues par d'autres tâches.

- soit lorsque l'accès aux données transférées par l'entrée-sortie est peu fréquent. Le programme émetteur de l'entrée-sortie peut libérer les ressources qu'il utilise, non liées à l'entrée-sortie, dès que le superviseur d'entrée-sortie lui a rendu le contrôle. Un mécanisme de protection doit cependant être mis en place pour que les données ne puissent être accédées tant que l'entrée-sortie n'est pas terminée.

  Exemple (figure 2.20) : dès qu'un fichier du disque système est fermé, le catalogue des fichiers de ce disque, bien qu'entièrement contenu en mémoire principale, est écrit sur le disque pour assurer le minimum de gravité en cas d'arrêt imprévu (par exemple coupure de courant).

Le programme qui fait cette sauvegarde en appelant le superviseur d'entrée-sortie utilise deux zones en mémoire principale :

- Une zone dans laquelle sont sauvegardés les registres à l'entrée dans le superviseur d'entrée-sortie. Cette zone n'est pas liée au transfert de données.
- Une zone contenant le descripteur de tâche d'entrée-sortie.

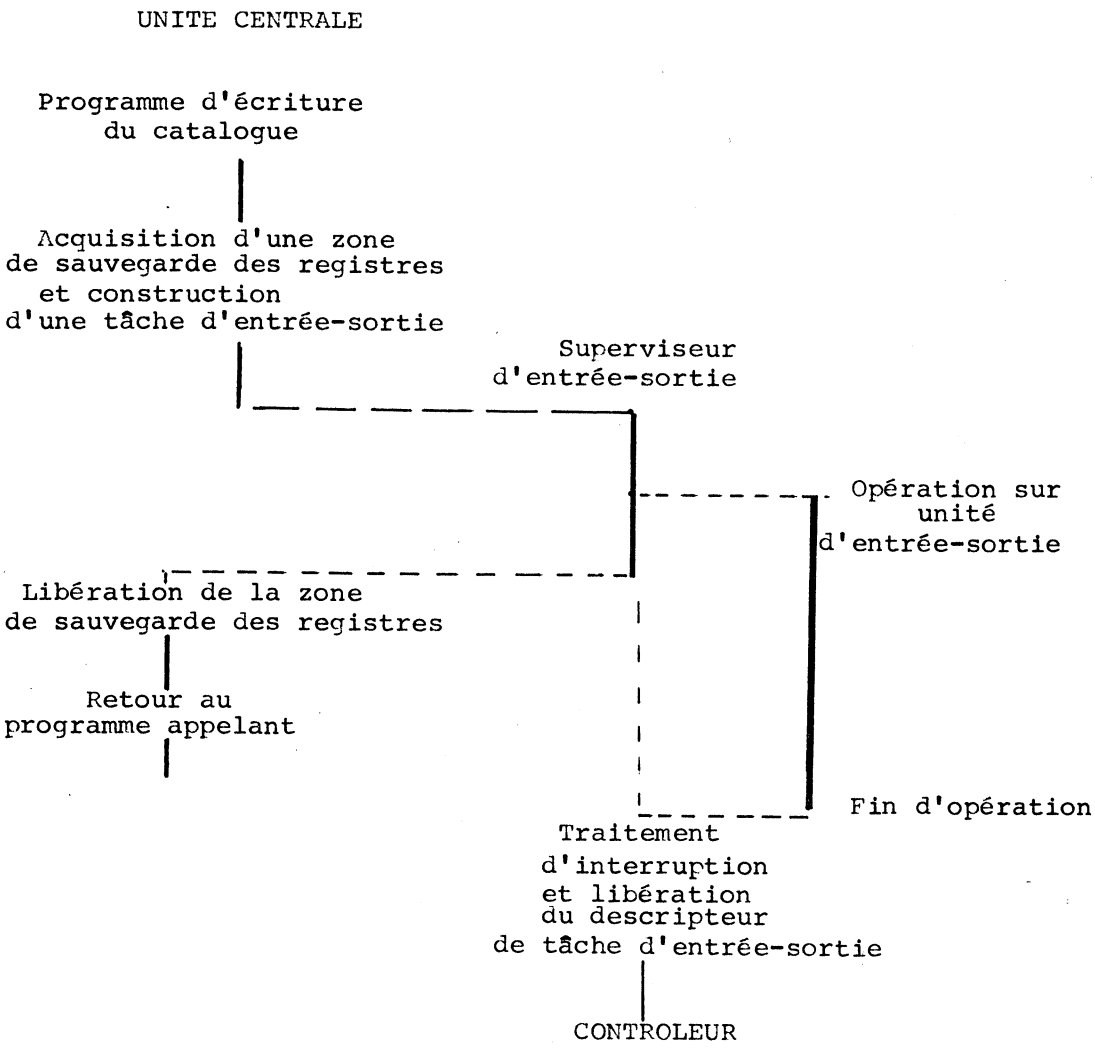


Figure 2.20 Exemple de tâche asynchrone

La première zone peut être libérée dès que le superviseur d'entrée-sortie a pris en compte la tâche d'entrée-sortie. Le descripteur de tâche d'entrée-sortie ne peut pas être libéré avant que l'entrée-sortie ne soit achevée.

Tous les renseignements permettant ainsi de synchroniser une tâche système et une entrée-sortie se trouvent dans le descripteur de tâche d'entrée-sortie.

Le bit WAITIO vaut 1 dans le cas d'une entrée-sortie synchrone, et vaut 0 dans le cas contraire.

Le deuxième mot du descripteur est un pointeur vers une zone dans laquelle est sauvegardé le contexte du processus émetteur, quand il est mis en sommeil dans l'attente de la fin d'entrée-sortie (entrée-sortie synchrone). Ce contexte se limite aux 16 registres tels qu'ils étaient lors de l'appel au superviseur d'entrée-sortie. Réactiver le programme revient à recharger les registres de l'ordinateur avec ces valeurs sauvegardées, et à se brancher au point de retour dont l'adresse est dans le registre 14. La réactivation des programmes est à la charge du contrôleur (2.1).

Le bit TDELE du deuxième mot indique au contrôleur si le descripteur de tâche doit être rendu à la mémoire libre du système lors de la réactivation du programme appelant. Cette option est choisie lorsque le programme émetteur de la demande d'entrée-sortie ne veut pas exploiter les renseignements qui lui sont fournis par le superviseur d'entrée-sortie.

Le quatrième mot RTND contient l'adresse d'une procédure appelée quand l'entrée-sortie est terminée. Cette procédure étant spécifiée par le programme émetteur de la demande d'entrée-sortie, elle sera chargée de libérer des ressources occupées pour l'entrée-sortie (telles que zones de mémoire acquises dynamiquement) ou de réveiller

des tâches système mises en sommeil en attendant la fin de cette entrée-sortie. Cette procédure est le seul moyen laissé à une tâche système d'intervenir en fin d'entrée-sortie, pour une opération asynchrone. Un indicateur DELETE indique au superviseur d'entrée-sortie si le descripteur de tâche doit être rendu à la mémoire libre du système au retour de cette procédure.

2.2222 Tâches auto-correctrices SCR

Une autre option permet au programme émetteur de modifier le cheminement dans le superviseur d'entrée-sortie : c'est l'option SCR (self completing request : tâche auto-correctrice). GMS est un hyperviseur qui doit générer des ressources virtuelles pour plusieurs copies du système CMS. La solution adoptée pour les disques et les bandes a été d'associer à une unité virtuelle une unité réelle de même type. Le système CMS ayant déjà ses procédures de traitement d'interruptions sur ces unités, il n'est pas nécessaire de refaire ce traitement d'interruptions dans le superviseur d'entrée-sortie de GMS. Pour les entrées-sorties lancées par CMS sur ces unités, on utilise cette option SCR : quand une interruption pour une telle entrée-sortie se produit, elle n'est pas analysée par GMS, mais on indique que la tâche d'entrée-sortie est terminée, et la tâche système qui a émis l'entrée-sortie est réveillée. Elle transmet le résultat de l'entrée-sortie à CMS qui traite et analyse l'interruption.

2.2223 Programme canal ou demande symbolique

Un programme peut faire une demande d'entrée-sortie sous forme symbolique ou sous forme d'un programme canal.

Une demande d'entrée-sortie sous forme symbolique est représentée par l'identification des données à transférer (adresse, longueur), et d'éventuelles options. Une demande est faite sous cette forme lorsque le programme canal pour cette entrée-sortie dépend de l'état dans lequel se trouve l'unité quand l'instruction SIO est exécutée (par exemple sur les terminaux 2741). C'est le superviseur d'entrée-sortie qui se charge de construire le programme canal avant de lancer le SIO.

L'adresse du programme canal, qu'il soit construit par le programme appelant ou le superviseur d'entrée-sortie, est conservée dans la zone ADPROGCAN du descripteur de tâche. Avec cette adresse, on trouve la clé de protection mémoire appliquée à l'opération d'entrée-sortie. Cette clé est utilisée pour les entrées-sorties qui ne sont pas demandées par GMS, donc qu'il faut empêcher de modifier des zones réservées à GMS. Les programmes canaux soumis par CMS et concernant les disques et les bandes, n'étant pas analysés par GMS, sont soumis à cette protection.

2.2224 Mot d'état canal

On trouve encore dans le descripteur le mot d'état canal CSW donné par l'interruption d'entrée-sortie correspondant à l'opération, ainsi que, éventuellement, le CSW correspondant à l'opération d'analyse qui a suivi et les données d'analyse. Toutes ces informations peuvent être récupérées par le programme émetteur de l'entrée-sortie.

2.223 Soumission de la tâche d'entrée-sortie au superviseur d'entrée-sortie

Le rôle du superviseur d'entrée-sortie lorsqu'il est activé par appel par une tâche système, est d'initialiser l'entrée-sortie demandée si le chemin menant à l'unité est libre et de chaîner le descripteur de tâche d'entrée-sortie au descripteur d'unité (figure 2.21). Il est appelé par une instruction de la forme

CALL IOSUP (A, B) ;

A est l'adresse du descripteur de l'unité sur laquelle doit se faire l'entrée-sortie, et B est l'adresse du descripteur de la tâche d'entrée-sortie.

Il faut d'abord retrouver dans l'arbre qui décrit la configuration du système (2.21) le chemin qui mène au descripteur de l'unité visée. Dans le descripteur de l'unité, se trouve son adresse physique. Par cette adresse, on retrouve les adresses des descripteurs du canal, et de l'unité de contrôle desservant l'unité, ainsi que l'adresse de son premier descripteur. On détermine ainsi si le canal et l'unité de contrôle sont libres : en cas d'occupation, l'indicateur correspondant du descripteur a la valeur 1. Pour que l'entrée-sortie puisse être initialisée, il faut en outre que l'unité concernée soit disponible : il faut donc que l'indicateur "OCCUPE" dans le descripteur de l'unité soit nul, et que l'unité soit prête (indicateur "NON PRET" nul dans le descripteur d'unité). Si l'opération d'entrée-sortie ne peut être initialisée pour une de ces raisons, la tâche d'entrée-sortie est rajoutée en queue de la file d'attente active . On se branche ensuite à la séquence commune pour la sauvegarde du contexte (registres).

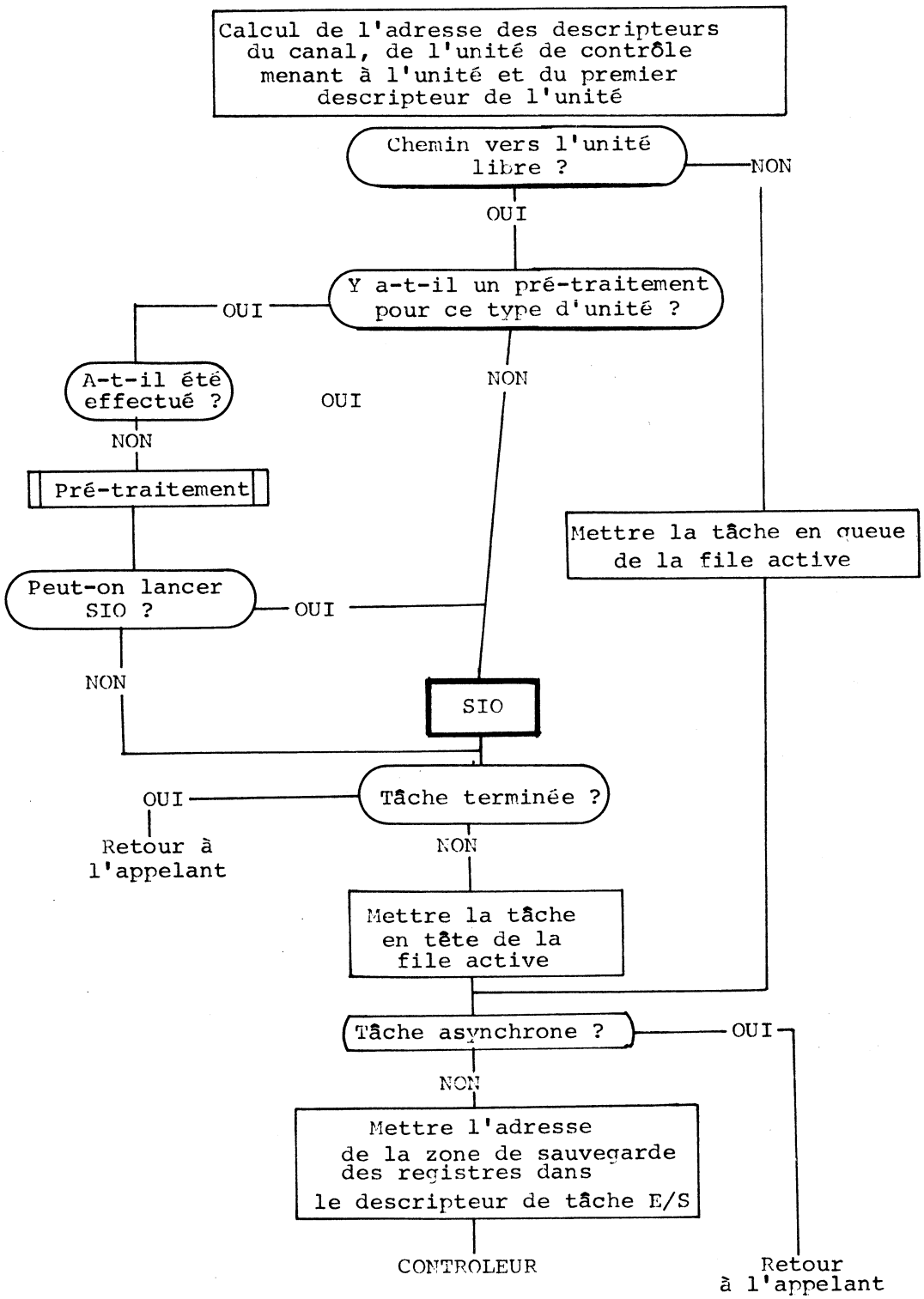


Figure 2.21 Traitement d'une tâche d'entrée-sortie sur entrée par appel

Si l'opération d'entrée-sortie peut être initialisée, il peut y avoir un pré-traitement à faire avant de lancer l'entrée-sortie. Ce pré-traitement est indépendant de l'appelant, il dépend seulement du type d'unité. Pour la configuration dont nous disposons, ce pré-traitement est employé pour les disques et pour les terminaux 2741. Pour les disques, il consiste à positionner le bras supportant les têtes de lecture-écriture, si l'entrée-sortie demande ce déplacement. Dans l'affirmative, cet ordre de positionnement est exécuté isolément. Le canal est ainsi libéré très rapidement et reste disponible pour une opération sur une autre unité. L'opération d'entrée-sortie demandée par le programme appelant ne sera lancée que lorsque le bras sera positionné.

Le pré-traitement, sur un terminal 2741, est dû à deux raisons. Il n'existe pas, sur un terminal 2741, de dispositif permettant à l'utilisateur de créer une interruption dans le système si une opération d'entrée-sortie n'est pas en cours sur le terminal. Une telle interruption amène généralement le système à entamer un dialogue avec l'utilisateur. Pour pallier cette absence, on peut exécuter une commande canal "Prepare" qui établit une connexion permanente entre le terminal et l'ordinateur. Cette commande permet à l'utilisateur de créer une interruption même si le clavier du terminal est bloqué : il lui suffit d'appuyer sur la touche "ATTENTION", une interruption est alors présentée au système. Mais une écriture ou une lecture peut être demandée au système sans que l'utilisateur n'appuie sur la touche "ATTENTION". La commande "Prepare" mobilisant l'unité pour une durée infinie, il faut l'arrêter pour pouvoir lancer la commande d'écriture ou de lecture. C'est donc dans le pré-traitement qu'une instruction HIO est émise pour libérer l'unité. La libération de l'unité sera indiquée par une interruption.

L'autre rôle du pré-traitement pour les terminaux 2741 est de transformer une requête symbolique d'entrée-sortie en un programme canal.

Les adresses des procédures de pré-traitement se trouvent dans une table. Les unités de la configuration ont été classées par type ; à chaque type correspond un index dans la table (figure 2.22). Le type de chaque unité est contenu dans son descripteur. Nous avons défini quatre types d'unités :

-Disques

-Bandes

-Lecteur-perforateur de cartes, imprimante, console opérateur 1052

-Terminaux 2741

Une valeur nulle dans la table indique qu'il n'y a pas de procédure de pré-traitement.

Dans certains cas, l'opération d'entrée-sortie peut être lancée immédiatement après l'appel de la procédure de pré-traitement, par exemple si le bras d'un disque est déjà positionné. Dans les autres cas, il faut attendre une interruption. Aussi la procédure de pré-traitement retourne un code au superviseur d'entrée-sortie : ce code est nul si et seulement si l'entrée-sortie peut être lancée. Sinon, il y a branchement immédiat à la séquence de mise en queue sur l'unité : l'opération sera lancée sur interruption indiquant la fin du pré-traitement.

Déplacement		
0	Adresse de PRIO2314	Disques
4	0	Bandes
8	0	Lecteur-perforateur, imprimante,console
12	Adresse de PRIO2741	Terminaux

Table des procédures de pré-traitement

Figure 2.22

Ces procédures positionnent dans le descripteur de tâche d'entrée-sortie l'indicateur PRSIO lorsque le pré-traitement est arrivé à un stade tel que l'entrée-sortie pourrait être initialisée (par exemple le programme canal pour 2741 a été construit).

Lancement de l'entrée-sortie

Après l'appel de la procédure de pré-traitement, si elle a retourné un code nul, le superviseur d'entrée-sortie lance l'opération : l'adresse physique de l'unité est trouvée dans son descripteur, et la valeur à charger dans le mot d'adresse canal se trouve dans le descripteur de tâche d'entrée-sortie. Il exécute une instruction SIO. Le code condition en indique le résultat :

La réponse à l'instruction SIO est "unité non opérationnelle" : la seule action possible est d'envoyer un message à l'opérateur.
La réponse est "canal occupé" : le SIO est réexécuté. On peut assurer que cette boucle n'est pas infinie, car il a été vérifié précedemment que le chemin vers l'unité était libre : l'occupation

du canal ne peut être qu'un état provisoire.
La réponse est "CSW rangé", on peut faire l'analyse de cette réponse en deux temps :

- Soit le bit "occupé" dans le mot d'état canal vaut 1. Si cette indication provient d'une condition d'interruption présentée par l'unité concernée, il suffit de relancer l'instruction SIO. Sinon, il faut attendre que la condition d'interruption disparaisse : la séquence de lancement est abandonnée, la condition d'interruption disparaîtra quand elle sera acceptée par le système.
- Soit le bit "occupé" dans le mot d'état canal est nul. On peut analyser le mot d'état canal comme s'il y avait eu une interruption. Les indicateurs d'occupation sont mis à jour, et les procédures spécialisées spécifiées dans le descripteur de l'unité peuvent être appelées dans les mêmes conditions que sur interruption (2.233 et 2.234). Après ce traitement, la tâche est mise en queue sur l'unité à moins qu'elle ne soit terminée (par exemple sur un lecteur de cartes quand il n'y a plus de carte dans le magasin).
- Soit la réponse au SIO est "opération bien partie". si le canal utilisé est sélecteur, il est marqué "occupé" (s'il est multiplex, il sera toujours disponible pour une opération sur une autre unité).
  L'unité est aussi marquée "occupé".

Le contrôle doit maintenant être rendu à la séquence qui a demandé le lancement d'une entrée-sortie. Si cette demande était consécutive à une interruption, la séquence "relance activité canal" (2.235) reprend le contrôle. Sinon, on passe à la séquence ci-dessous.

Mise en file d'attente de la tâche d'entrée-sortie

Si la tâche a déjà été marquée "terminée" (RCOMPL=1) il n'y a pas lieu de la mettre dans une file d'attente. Le contrôle est immédiatement rendu à l'appelant.
Sinon, elle est mise en tête de la file des tâches d'entrée-sortie en attente sur l'unité. La tête de cette file est toujours la tâche active sur l'unité, ou la première tâche à activer si l'unité n'est pas occupée.

Si la tâche est asynchrone, le contrôle peut être immédiatement rendu à l'appelant.

Si la tâche est synchrone, il faut sauvegarder le contexte du programme appelant pour qu'il puisse être réactivé quand l'entrée-sortie sera terminée. Le programme appelant est une tâche système dont le contexte est constitué par les 16 registres de l'ordinateur. Ces registres ont été sauvegardés à l'entrée du superviseur d'entrée-sortie, suivant les conventions d'appel des procédures dans le système. Il suffit donc de mettre l'adresse de cet emplacement de sauvegarde dans le descripteur de tâche d'entrée-sortie. Cette tâche joue ainsi deux rôles : représenter, pour le superviseur d'entrée-sortie, une demande d'entrée-sortie, et représenter, pour le contrôleur, la tâche système qui a émis cette demande . Le contrôle est ensuite passé au contrôleur, pour qu'il active une autre tâche système.

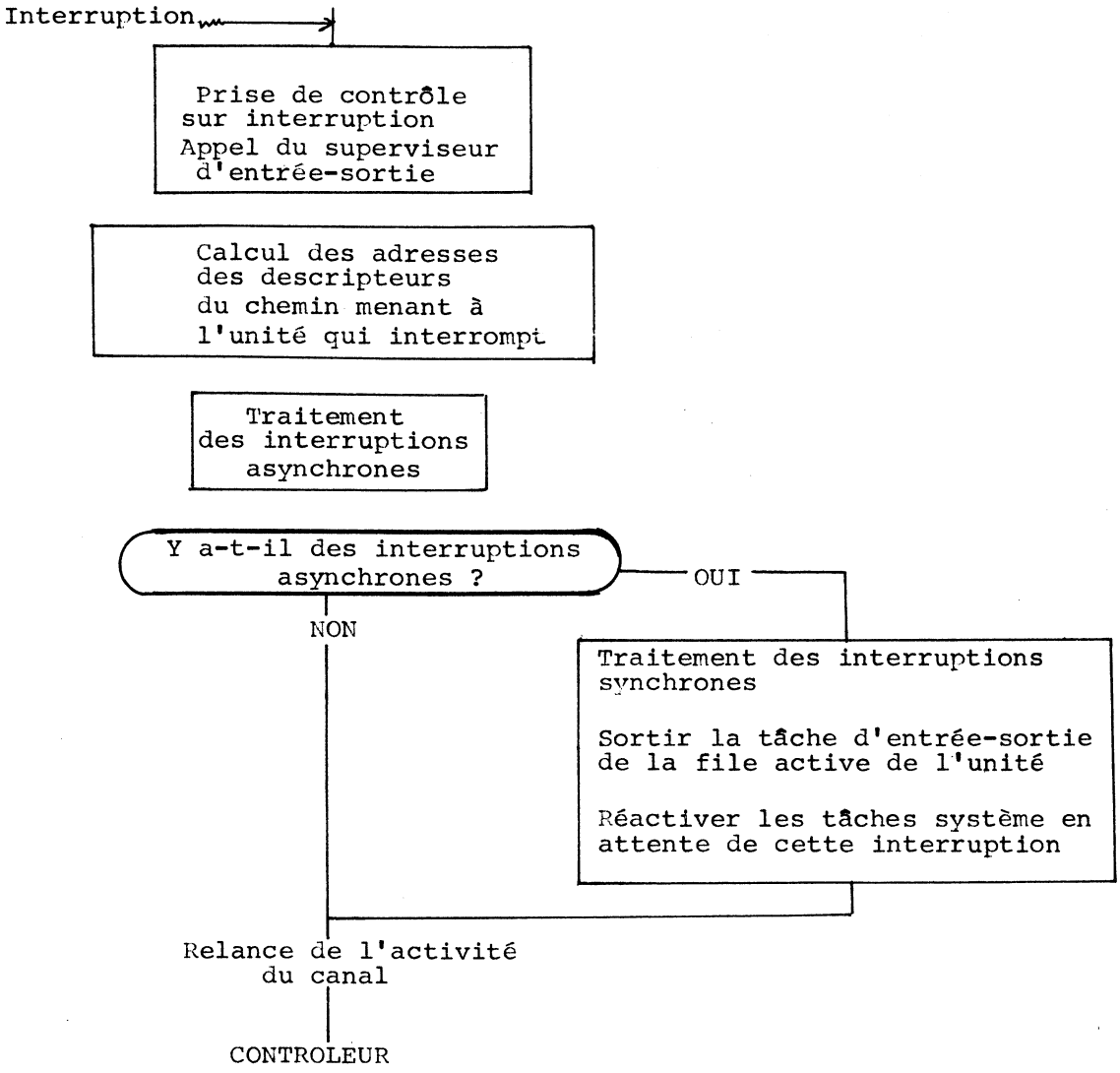
2.23 Activation du superviseur d'entrée-sortie sur interruption

Le traitement des interruptions d'entrée-sortie est le rôle du superviseur d'entrée-sortie. Une interruption marque la fin d'une opération d'entrée-sortie ou la modification de l'état d'une unité (passage de l'état "non prêt" à l'état "prêt"). Le superviseur d'entrée-sortie, étant chargé de la gestion des unités d'entrée-sortie, doit analyser ces interruptions. Il corrige les erreurs, détermine quand une opération d'entrée-sortie est achevée : pour cela, il peut appeler des procédures spécialisées par type d'unité. Il fait réactiver les tâches système qui attendent la fin d'une entrée-sortie. Enfin, il doit relancer des opérations sur les unités devenues accessibles par suite de l'interruption, en respectant la règle qu'il faut utiliser au mieux ces unités. Les différentes étapes suivies lors d'une interruption sont indiquées sur l'organigramme de la figure 2.23.

2.231 Prise de contrôle sur interruption

Les interruptions d'entrée-sortie doivent être traitées avec une priorité élevée. Elles impliquent généralement la libération d'une unité d'entrée-sortie, et il importe d'occuper rapidement ce qui vient d'être libéré. D'autre part, une interruption d'entrée-sortie peut amener une modification dans l'ordre de priorité des tâches demandant l'allocation de l'unité centrale (à cause du réveil d'une tâche système, ou de la création d'une nouvelle tâche système).

Aussi toute interruption d'entrée-sortie qui se présente est analysée immédiatement.

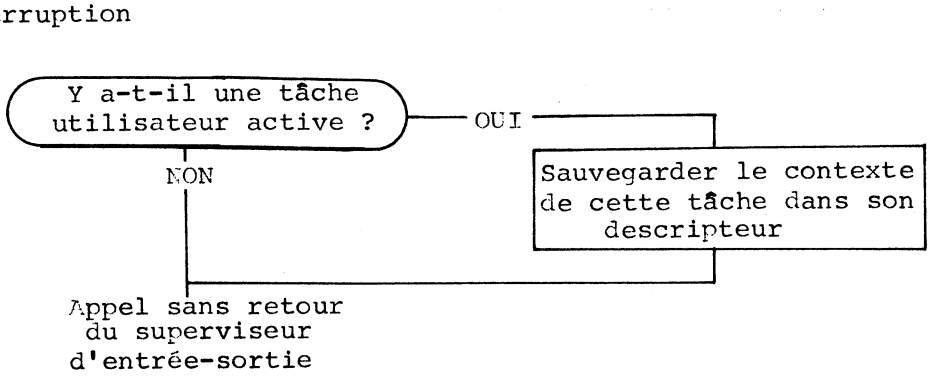


Traitement des interruptions
par le superviseur d'entrée-sortie

Figure 2.23

Sur interruption d'entrée-sortie, le contrôle est d'abord donné au programme IOINT (figure 2.24). Il sauvegarde le contexte de la tâche utilisateur active, s'il y en a une ; le contexte de cette tâche consiste en registres + PSW au moment de l'interruption. Puis il effectue un branchement direct au point d'entrée sur interruption du superviseur d'entrée-sortie.

Il serait bien possible d'intégrer ce programme dans le superviseur d'entrée-sortie. Mais cela contreviendrait au principe qu'il est chargé uniquement de gérer les unités d'entrée-sortie.



Traitement initial d'une interruption d'entrée-sortie

Figure 2.24

2.232 Calcul de l'adresse des descripteurs

A l'entrée dans le superviseur d'entrée-sortie, la seule indication connue concernant l'unité qui cause l'interruption est son adresse physique "abc", qui se trouve dans le code interruption de l'ancien mot d'état programme, à l'adresse 56 en mémoire basse. A partir de cette adresse, on retrouve, par indexation dans l'arbre qui décrit la configuration, les adresses des descripteurs du canal, de l'unité de contrôle et de l'unité qui crée l'interruption (2.21). Cette recherche est semblable à celle qui est faite à l'entrée du superviseur d'entrée-sortie, lorsqu'il est appelé par un programme qui lui soumet une tâche d'entrée-sortie (2.223). La seule différence est que, sur interruption, on ne dispose pas de l'adresse du descripteur d'unité, alors qu'elle est fournie en paramètre lors d'un appel. Si l'unité qui interrompt ne figure pas dans les tables (par exemple unité non connue à la génération du système),

l'opérateur du système est prévenu, et le traitement de l'interruption s'arrête ici.

2.233 Traitement des interruptions asynchrones

Une interruption asynchrone est une interruption qui n'est pas consécutive au lancement d'une tâche d'entrée-sortie connue dans le système sous la forme d'un descripteur. Ces interruptions peuvent se produire dans les cas suivants :

1- L'unité passe de l'état "non prêt" à l'état "prêt" par intervention de l'opérateur (disques, bandes, lecteur, perforateur, imprimante). L'interruption "fin sur unité" est alors automatiquement présentée.

2- L'opérateur appuie sur la touche "ATTENTION" de la console 1052.

3- Un utilisateur appuie sur la touche "ATTENTION" de son terminal après qu'une commande "Prepare" ait été lancée ; une commande "Prepare" n'est pas représentée par une tâche d'entrée-sortie puisque son but est précisement de permettre les interruptions asynchrones.

4- Une instruction HIO a été émise pour arrêter une opération sur une unité.

Le premier cas peut survenir dans deux situations (figure 2.25) :

- des tâches d'entrée-sortie sont chaînées au descripteur de l'unité, et on attend que l'unité soit prête pour lancer la première. Il suffit d'indiquer dans le descripteur d'unité dès qu'elle est prête (bit DVNR nul). Le passage automatique dans la partie "relance activité canal" (2.235) suffira à relancer cette tâche.

- aucune tâche d'entrée-sortie n'est chaînée au descripteur de l'unité. On peut considérer que, dans ce cas, c'est un moyen accordé à l'opérateur ou à un utilisateur pour faire exécuter un programme du système : par exemple, quand le lecteur de cartes passe dans l'état "prêt", il faut créer une tâche système pour lire les cartes. On se trouve dans la même situation que dans les cas 2 et 3 ci- dessus. Toutefois, il ne faut pas que cette tâche système interrompe pendant un temps indéterminé le superviseur d'entrée-sortie qui doit gérer de manière optimale toutes les unités. Aussi l'activation de cette tâche système est faite en deux étapes. Dans la première étape, le superviseur d'entrée-sortie appelle une procédure ATTNPR dont l'adresse se trouve dans le descripteur de l'unité. Cette procédure connaît la tâche système à créer. Elle acquiert un descripteur de tâche système, qu'elle initialise avec les bons paramètres et la met dans la file des tâches système à activer par le contrôleur. La deuxième étape sera l'activation de cette tâche par le contrôleur (figure 2.26).

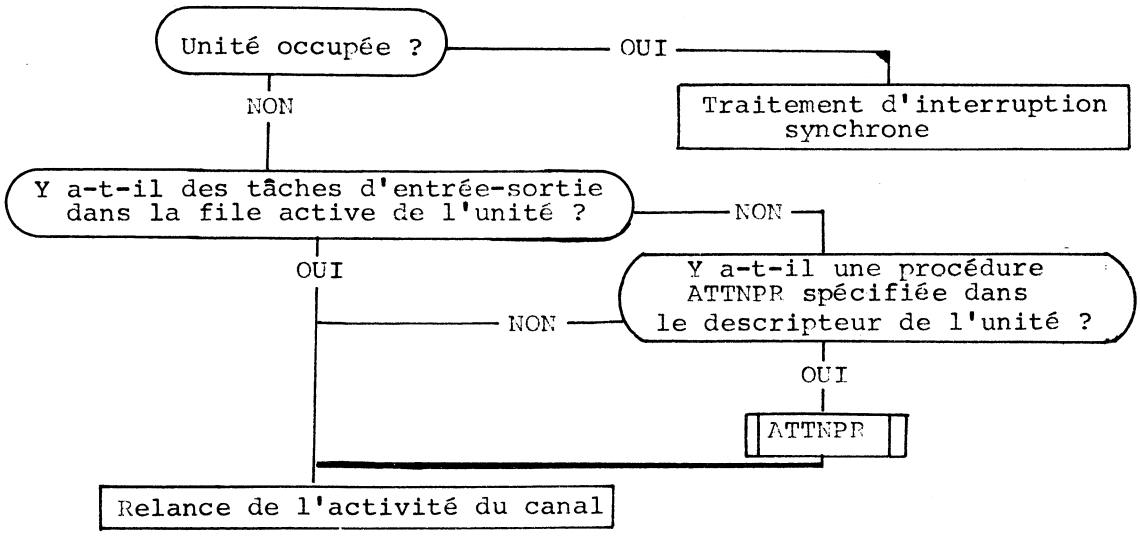
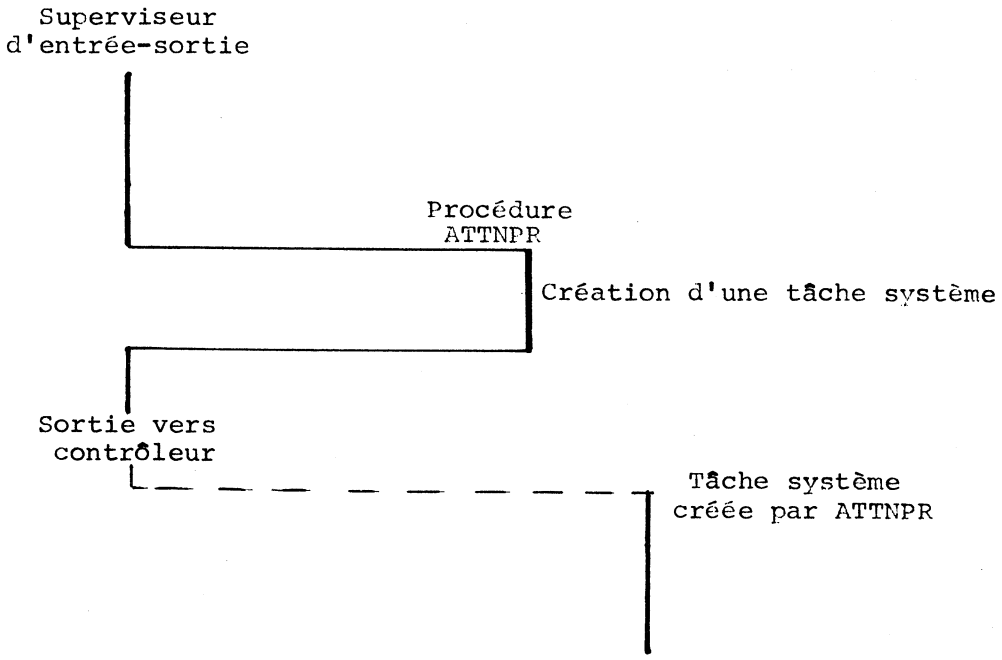


Figure 2.25 Traitement d'interruption synchrone "Fin sur unité"



Création et activation
d'une tâche système
sur interruption asynchrone

Figure 2.26

La procédure ATTNPR après avoir créé la tâche système rend immédiatement le contrôle au superviseur d'entrée-sortie.

Si l'interruption asynchrone s'est produite en même temps qu'une interruption synchrone, cette dernière est maintenant analysée (2.234). Dans les autres cas, la sequence "relance activité canal" est immédiatement activée.

Le quatrième cas est celui d'une interruption créée par l'arrêt d'une opération d'entrée-sortie par HIO. L'arrêt de l'opération est indiqué dans le descripteur d'unité, et la séquence de relance de l'activité du canal est activée.

2.234 Traitement des interruptions synchrones

Les interruptions traitées dans cette partie sont relatives à une tâche d'entrée-sortie. La tâche d'entrée-sortie active sur cette unité est toujours celle qui se trouve en tête de la file active attachée à l'unité. C'est dans son descripteur que le superviseur d'entrée-sortie trouve les informations sur l'entrée-sortie qui vient de se terminer, et qu'il range les renseignements amenés par l'interruption. Cette séquence est détaillée dans l'organigramme de la figure 2.27.

Il faut d'abord remarquer qu'une tâche d'entrée-sortie peut donner lieu au lancement d'une autre opération d'entrée-sortie par le superviseur d'entrée-sortie, par exemple quand une opération se termine par "erreur sur unité". Dans ce cas, le superviseur d'entrée-sortie lance automatiquement une opération d'analyse apportant d'autres informations sur l'erreur. L'interruption terminant cette analyse est synchrone puisqu'elle est associée à une tâche d'entrée-sortie. Toutefois elle subit un traitement initialement différent de celui d'une interruption due à la tâche d'entrée-sortie elle-même. Le bit SENSE dans le descripteur de l'unité a la valeur 1 quand une opération d'analyse a été déclenchée.

Si l'interruption indique la fin d'un positionnement de bras sur un disque, le chaînage de commandes est rétabli et on passe immédiatement à la relance de l'activité du canal : il n'y rien d'autre à faire, puisque c'est une opération supplémentaire faite par le superviseur d'entrée-sortie pour optimisation, et qu'elle est totalement inconnue en dehors du superviseur d'entrée-sortie.

Si l'interruption ne marque pas une fin d'analyse (bit SENSE nul), le mot d'état canal est sauvé dans le descripteur de la tâche d'entrée-sortie : il pourra être lu par le programme émetteur de la tâche d'entrée-sortie.

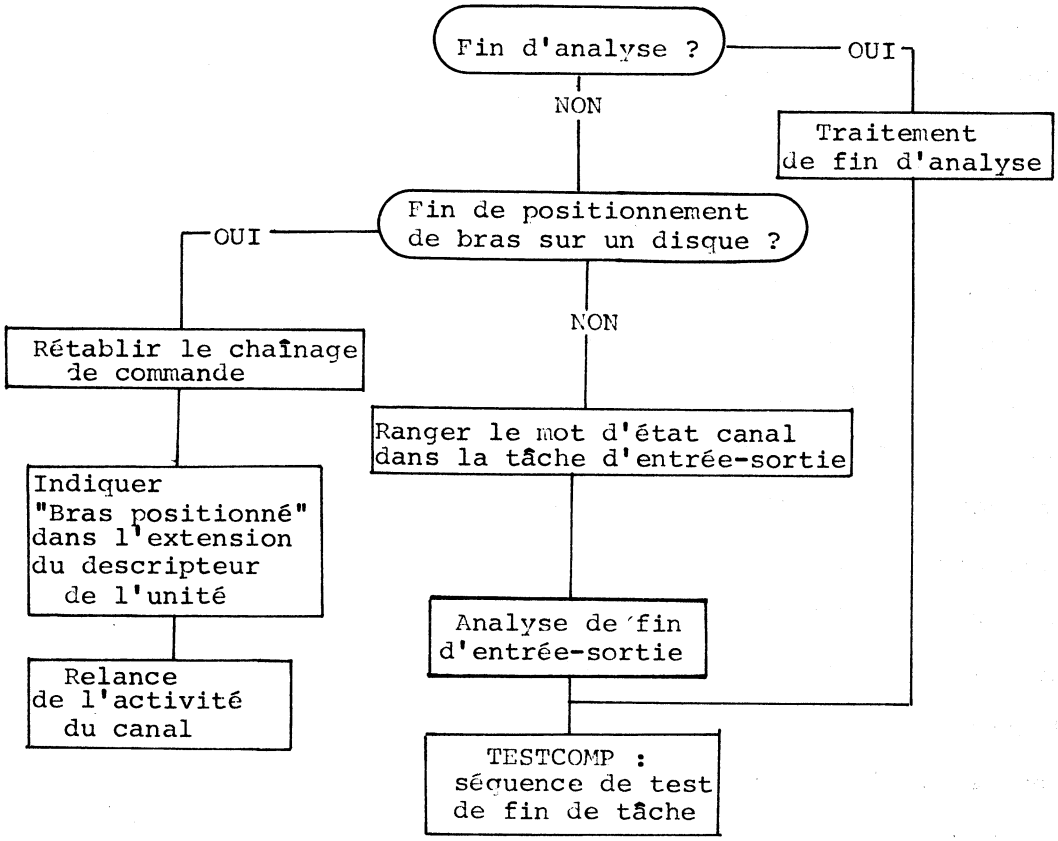


Figure 2.27 Traitement des interruptions synchrones

2.2341 Fin de tâche d'entrée-sortie

Toutes les interruptions traitées ici sont directement dûes à une tâche d'entrée-sortie. Les interruptions traitées sont "fin sur canal", "fin sur unité", "exception sur unité", et "erreur sur unité". Si aucune condition exceptionnelle ne se produit, une entrée-sortie est achevée quand elle a émis les interruptions "fin sur canal" et "fin sur unité". Si une entrée-sortie cause une interruption "erreur sur unité" ou "exception sur unité", il se peut que l'une ou les deux interruptions ci-dessus ne se produisent pas. Une entrée-sortie est donc considérée comme terminée si une des trois interruptions suivantes a eu lieu :

"Fin sur unité" ou "Exception sur unité" ou "Erreur sur unité"

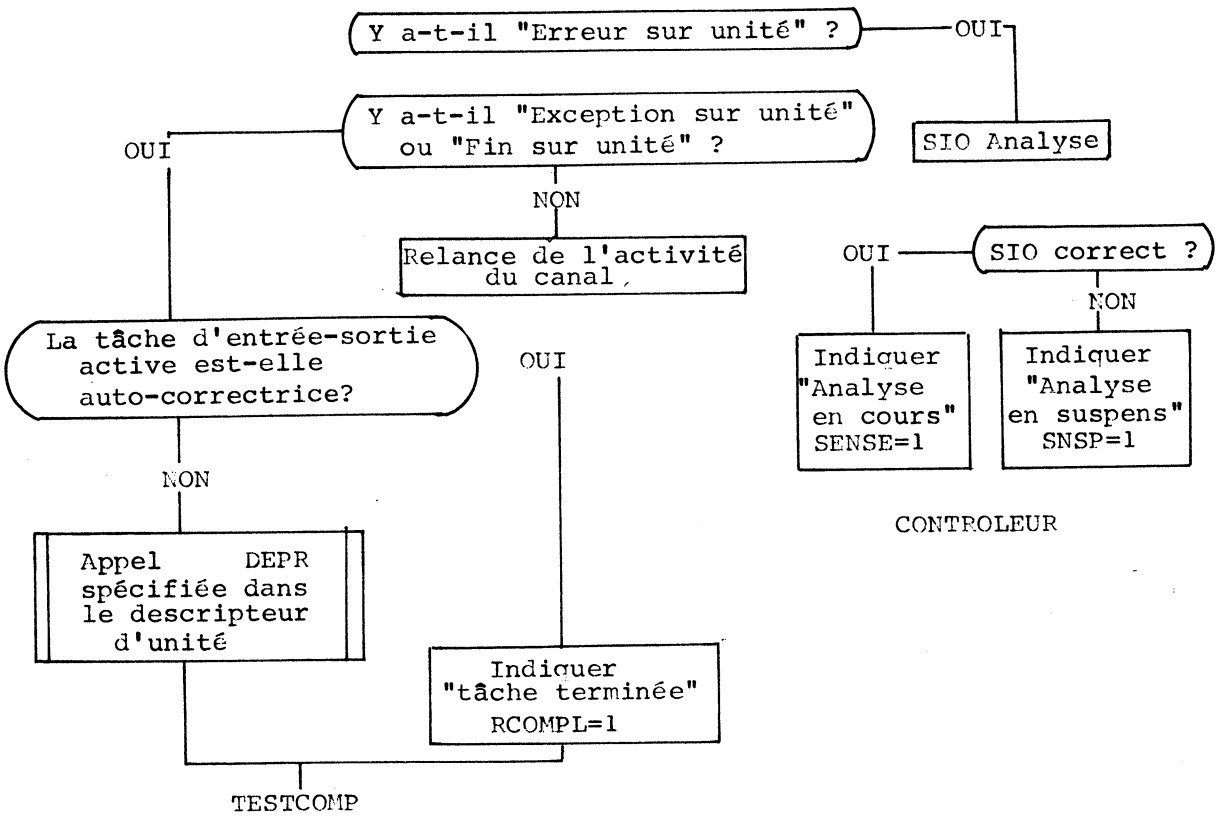


Figure 2.28 Traitement des interruptions directement dues à une tâche d'entrée-sortie

Si l'interruption indique seulement "fin sur canal", il faut attendre "fin sur unité". Aussi passe-t-on immédiatement à la séquence suivante TESTCOMP (2.2343) dans ce cas.

S'il y a "Exception sur unité" ou "Fin sur unité", l'entrée-sortie est physiquement terminée. Mais il se peut qu'au point de vue du système ou de l'utilisateur, elle ne soit pas achevée.

Si c'est une tâche de type SCR (2.222), on indique dans son descripteur qu'elle est terminée (RCOMPL=1). Le programme qui a émis cette demande analysera l'interruption et décidera éventuellement de relancer une demande d'entrée-sortie. Sinon, le superviseur d'entrée-sortie fait appel à une procédure spécifiée dans le descripteur de l'unité. Cette procédure détermine, en fonction de l'unité, la signification de l'interruption. Son rôle principal est de déterminer si la tâche d'entrée-sortie est achevée, ou si elle doit être relancée. La relance sera faite automatiquement par le superviseur d'entrée-sortie : il suffit que la tâche ne soit pas marquée "terminée".

Exemple : pour le lecteur de cartes, une interruption "Exception sur unité" indique que le paquet de cartes à lire est épuisé. La tâche d'entrée-sortie est alors marquée "terminée" et un indicateur positionné pour prévenir le programme de cet évènement particulier. Autre exemple, pour un terminal 2741, en lecture, l'utilisateur peut terminer la ligne en appuyant sur la touche "retour chariot" ou sur la touche "Attention" du terminal (figure 2.30). La convention adoptée dans GMS est qu'une ligne terminée par "Retour chariot" est transmise au programme qui a émis la tâche d'entrée-sortie, alors qu'une ligne terminée par "Attention" est annulée et qu'il faut relancer la lecture.

Une ligne ne comprenant que "Attention" entraîne l'appel de la procédure dont l'adresse se trouve dans la zone PR2741AT de la tâche d'entrée-sortie et un indicateur est positionné dans l'extension du descripteur de l'unité. Si la zone PR2741AT est nulle, c'est le programme appelant qui analysera l'"Attention". La figure 2.29 décrit l'organigramme de cette procédure.

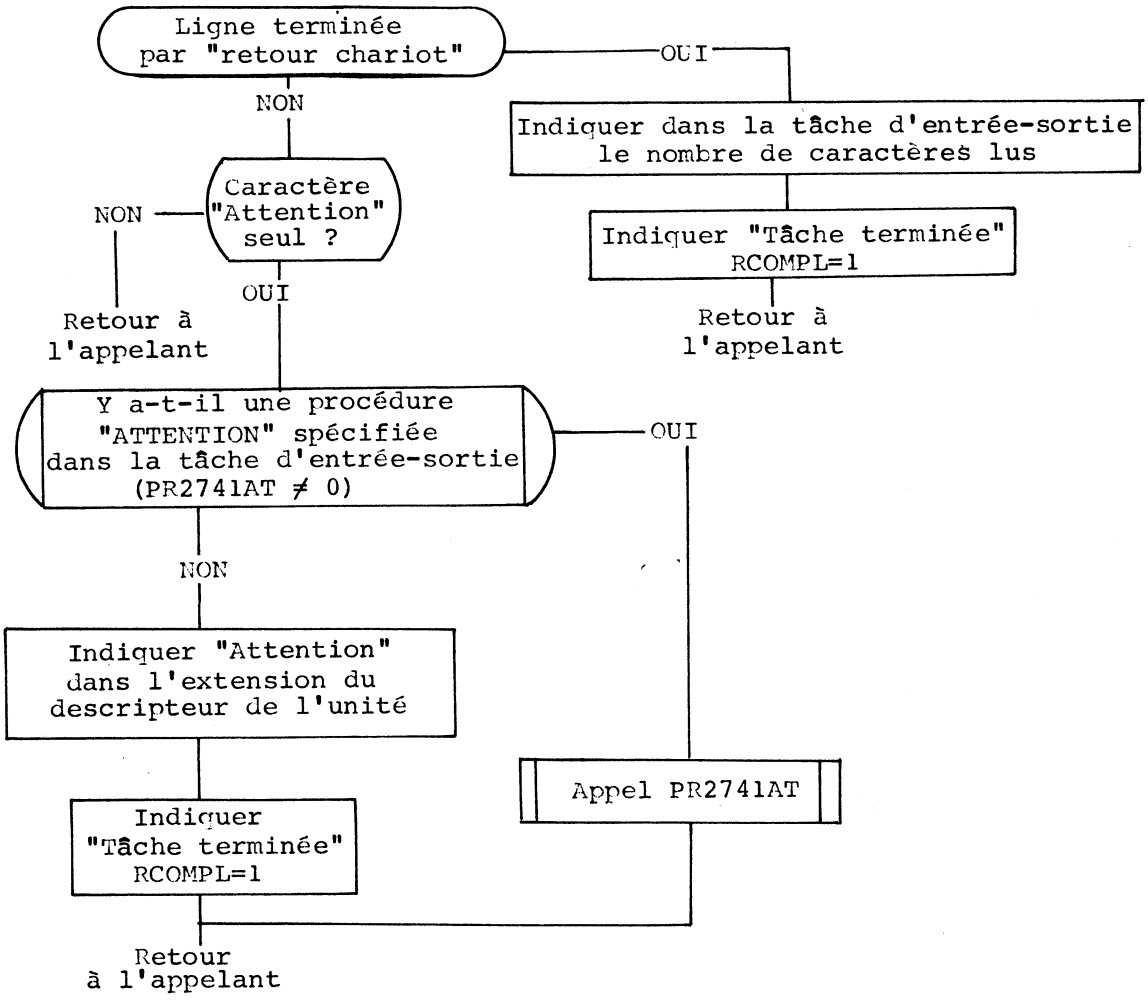


Figure 2.29 Traitement d'une fin de lecture par la procédure DEPR des terminaux

abbcA : ligne annulée

A : annulation de commande GMS
ou passage de CMS à GMS

abcdR : ligne correcte

La lettre A symbolise le caractère envoyé par la touche "ATTENTION"

La lettre R symbolise les caractères envoyés par la touche "Retour Chariot"

Les lettres minuscules représentent des caractères frappés par l'utilisateur

Figure 2.30 Conventions de frappe dans GMS

Si le mot d'état canal indique "Erreur sur unité", avant d'entreprendre une autre action, le superviseur d'entrée-sortie émet une commande d'analyse sur cette unité. Cette commande délivre une information précisant l'erreur (un à huit octets selon les unités). A moins que cet état d'erreur ne soit permanent (par exemple unité non prête), cette information est détruite dans les circuits de cablage dès qu'une nouvelle opération d'entrée-sortie est initialisée sur cette unité. Aussi l'opération d'analyse doit être lancée dès réception de "Erreur sur unité". Les octets d'analyse sont envoyés dans le descripteur de tâche d'entrée-sortie, pour pouvoir être transmis au programme appelant, s'il y a lieu. Si l'opération d'analyse ne peut démarrer (à cause d'une interruption en suspens), on indique "Analyse en suspens" dans le descripteur d'unité. L'analyse démarrera dès que l'interruption sera passée.

Autrement, on indique "Analyse en cours" dans le descripteur de l'unité. Dans les deux cas, la séquence "relance de l'activité canal" ne sera pas activée : on ne peut occuper le canal tant que l'analyse n'est pas terminée.

2.2342 Fin d'analyse

Le traitement de la fin d'une commande d'analyse se fait comme il est indiqué à la figure 2.31. Le traitement diffère suivant qu'il s'agit ou non d'une tâche auto-correctrice. Une tâche auto-correctrice laisse au programme émetteur de la tâche d'entrée-sortie le soin d'analyser l'erreur, toutes informations utiles ayant été mises dans le descripteur de tâche d'entrée-sortie. La tâche est immédiatement marquée "terminée".

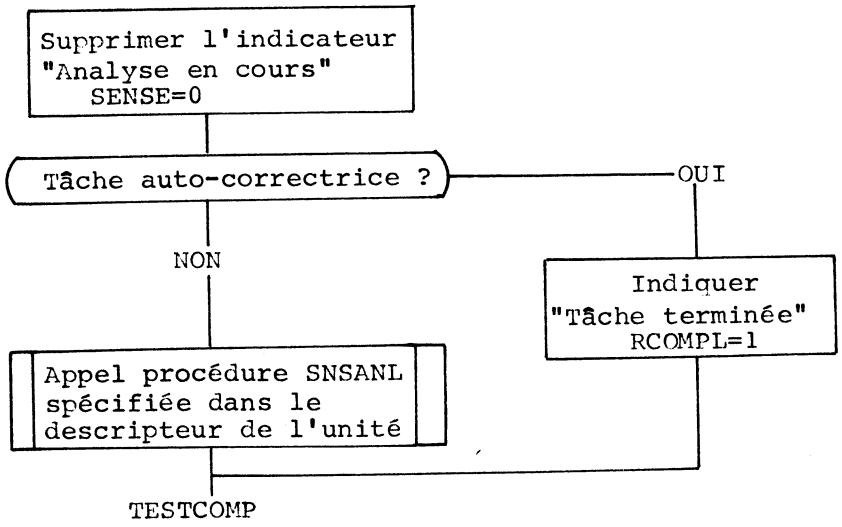


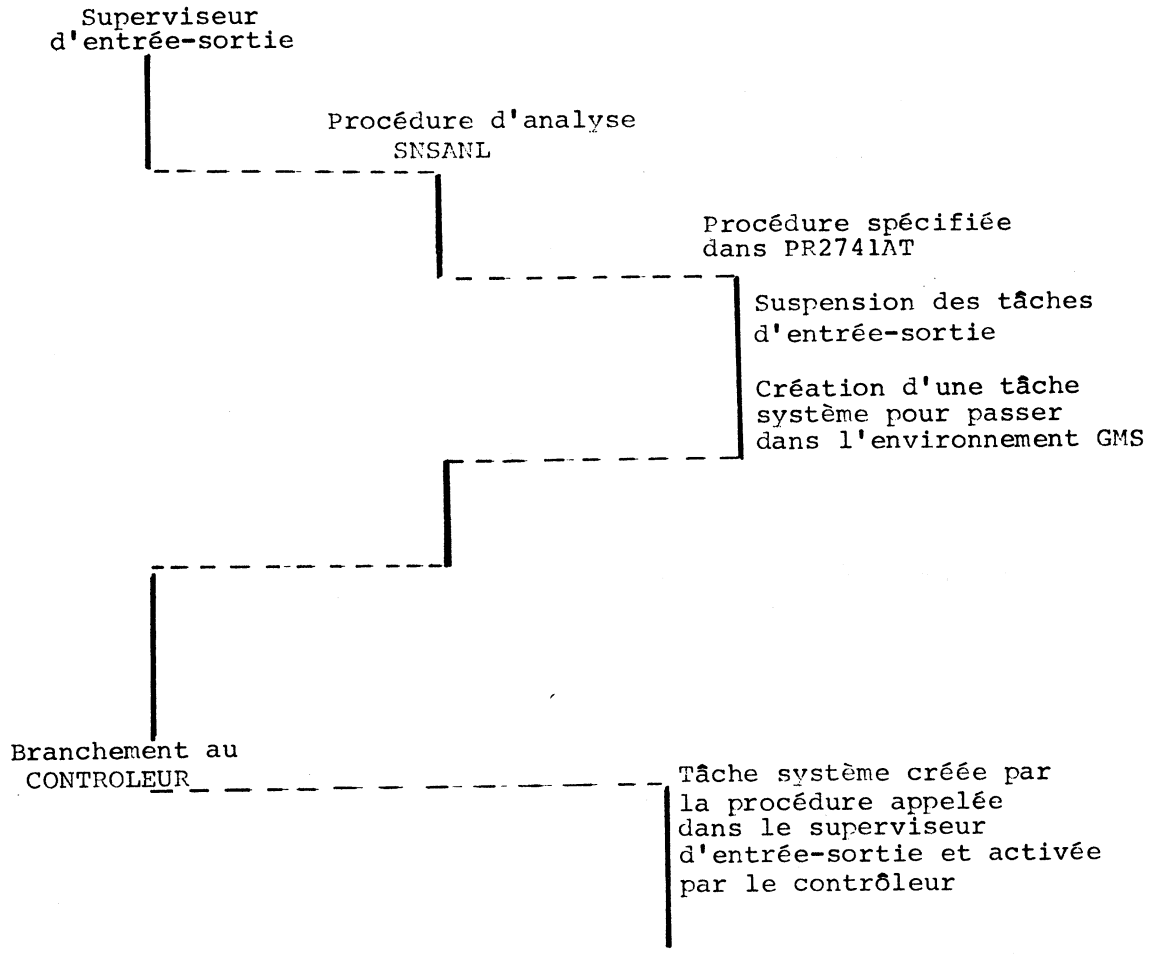
Figure 2.31 Traitement de fin d'analyse

Les tâches auto-correctrices ont été introduites pour traduire les demandes d'entrée-sortie de CMS sur disque et bande, ces unités étant dédiées à CMS. CMS ayant déjà un traitement d'erreur de ces unités, on évite ainsi d'analyser deux fois les erreurs, dans GMS puis dans CMS, sans avoir à modifier CMS. Un inconvénient de cette méthode est qu'elle implique un délai plus grand entre la détection d'une erreur et sa correction.

Si la tâche n'est pas auto-correctrice, le superviseur d'entrée-sortie appelle la procédure d'analyse dont l'adresse est spécifiée dans le descripteur d'unité. Cette procédure utilise les informations d'analyse qui sont dépendantes de l'unité. Elle prévient l'opérateur si son intervention est nécessaire, met à jour un compteur d'erreur. Si l'erreur n'est pas permanente, elle laisse le superviseur relancer l'entrée-sortie. Si l'erreur est permanente, elle indique dans le descripteur d'unité qu'elle n'est pas prête, et les tâches d'entrée-sortie sont suspendues tant que l'unité n'est pas prête.

Enfin, il se peut qu'il n'y ait pas une erreur véritable, mais que soit indiquée ainsi une terminaison anormale de l'entrée-sortie. Cela arrive sur un terminal 2741, quand l'utilisateur interrompt l'impression d'une ligne en appuyant sur la touche "Attention". Le choix de l'action à entreprendre n'est pas du ressort du superviseur d'entrée-sortie, puisque ce n'est pas une erreur de matériel. Il spécifie, dans l'extension du descripteur du terminal, un indicateur qui signale cet évènement. Le programme qui a émis cette tâche peut spécifier, dans la zone PR2741AT, l'adresse d'une procédure appelée quand cet evènement se produit. Elle consitue ainsi un point d'intervention du programme créateur de la tâche d'entrée-sortie dans le superviseur d'entrée-sortie. Elle est indispensable pour les sorties de ligne correspondant à une écriture de CMS qui sont asynchrones. Par convention, dans cette situation, l'utilisateur passe dans l'environnement GMS : toutes les tâches d'entrée-sortie de la file active sur l'unité sont suspendues. Elles sont transférées dans la file des tâches suspendues (pointée par FTPENDTK et LSPENDTK dans le descripteur du terminal). Ensuite, cette procédure crée et fait activer une tâche système qui créera le dialogue GMS-utilisateur (figure 2.32).

Si la sortie se produisait pour GMS, il n'y a pas de changement d'environnement à faire : la zone PR2741AT est donc nulle. Seul un indicateur est positionné dans l'extension du descripteur d'unité, et la tâche est marquée "terminée".



Traitement d'une interruption créée pendant une écriture sur terminal

Figure 2.32

TESTCOMP : séquence de test de fin de tâche

Cette séquence reçoit le contrôle quand une interruption synchrone a été traitée. Elle assure la liaison avec le programme qui a émis la tâche d'entrée-sortie active sur l'unité. Si la tâche n'a pas été marquée "terminée" lors d'une des séquences précédentes, le contrôle est immédiatement donné à la relance de l'activité sur le canal (2.235). La liaison avec le programme appelant n'a pas à être établie à ce moment.

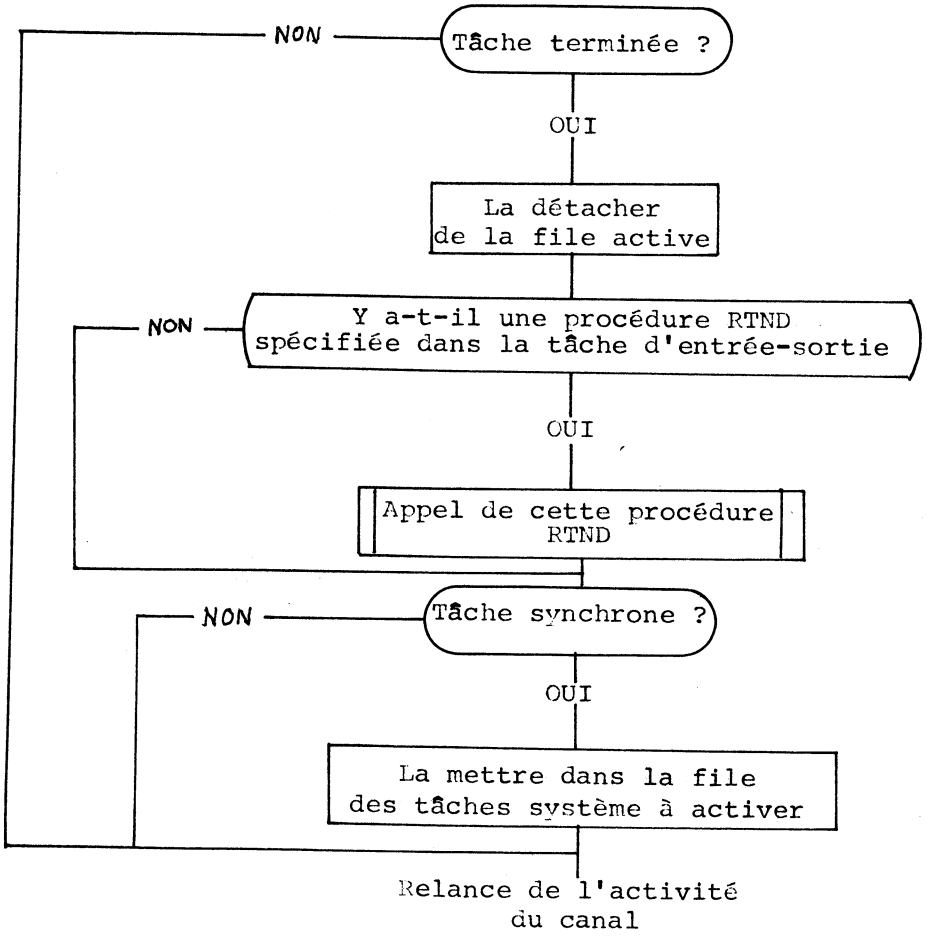


Figure 2.33 Séquence de test de fin de tâche

Si la tâche est terminée, la liaison de la fin d'entrée-sortie avec le programme appelant se fait de deux façons.

D'abord, la fin d'une entrée-sortie implique généralement des libérations de ressources (zones de mémoire, utilisation de programmes non réentrants, ...). Ces ressources varient suivant la tâche d'entrée-sortie. Aussi le programme qui a émis la demande d'entrée-sortie spécifie, s'il le désire, une adresse de procédure dans la zone RTND du descripteur de tâche d'entrée-sortie. Cette procédure est chargée de libérer les ressources mobilisées pour la tâche d'entrée-sortie. C'est aussi elle qui peut débloquer les tâches système attendant la libération d'une de ces ressources : enlever le verrou qui bloque l'accès à un catalogue, rendre disponible une zone tampon, ...

A la suite de cet appel de procédure, la tâche d'entrée-sortie est ôtée de la liste des tâches chaînées à l'unité. Si la tâche d'entrée-sortie est synchrone, il faut réveiller la tâche système qui l'a émise Pour cela, son descripteur est chaîné aux tâches système que le contrôleur doit activer.

2.235 Relance de l'activité canal

Le but de cette séquence est de profiter du changement d'état dans les unités pour relancer des opérations d'entrée-sortie en utilisant au mieux les possibilités de fonctionnement parallèle de ces unités. L'optimisation à faire se limite en fait aux unités de disque. Sur ces unités, une commande de lecture ou d'écriture est généralement précédée par un déplacement du bras qui supporte les têtes de lecture-écriture. Le lien entre ces commandes est fait par chaînage de commandes.

Or un canal sélecteur est occupé pendant le déroulement d'un programme canal, jusqu'à ce que la dernière commande canal ait libéré le canal. Une commande de positionnement du bras (commande SEEK) ne mobilise le canal que pendant le temps de transfert des données (c'est-à-dire le numéro du cylindre sur lequel il faut se positionner). Ensuite, le canal est libre s'il n'y a pas chaînage de commande, mais l'unité reste occupée pendant tout le déplacement du bras (de 25 à 100ms sur un 2314).

Pour améliorer la gestion du canal, le superviseur d'entrée-sortie explore en deux passages les descripteurs d'unités rattachées à un canal sélecteur : au premier passage, il essaie de lancer des opérations isolées de positionnement du bras sur toutes les unités de disque rattachées au canal. Le premier passage est indiqué par la valeur 1 du bit FSTPASS du descripteur du canal.

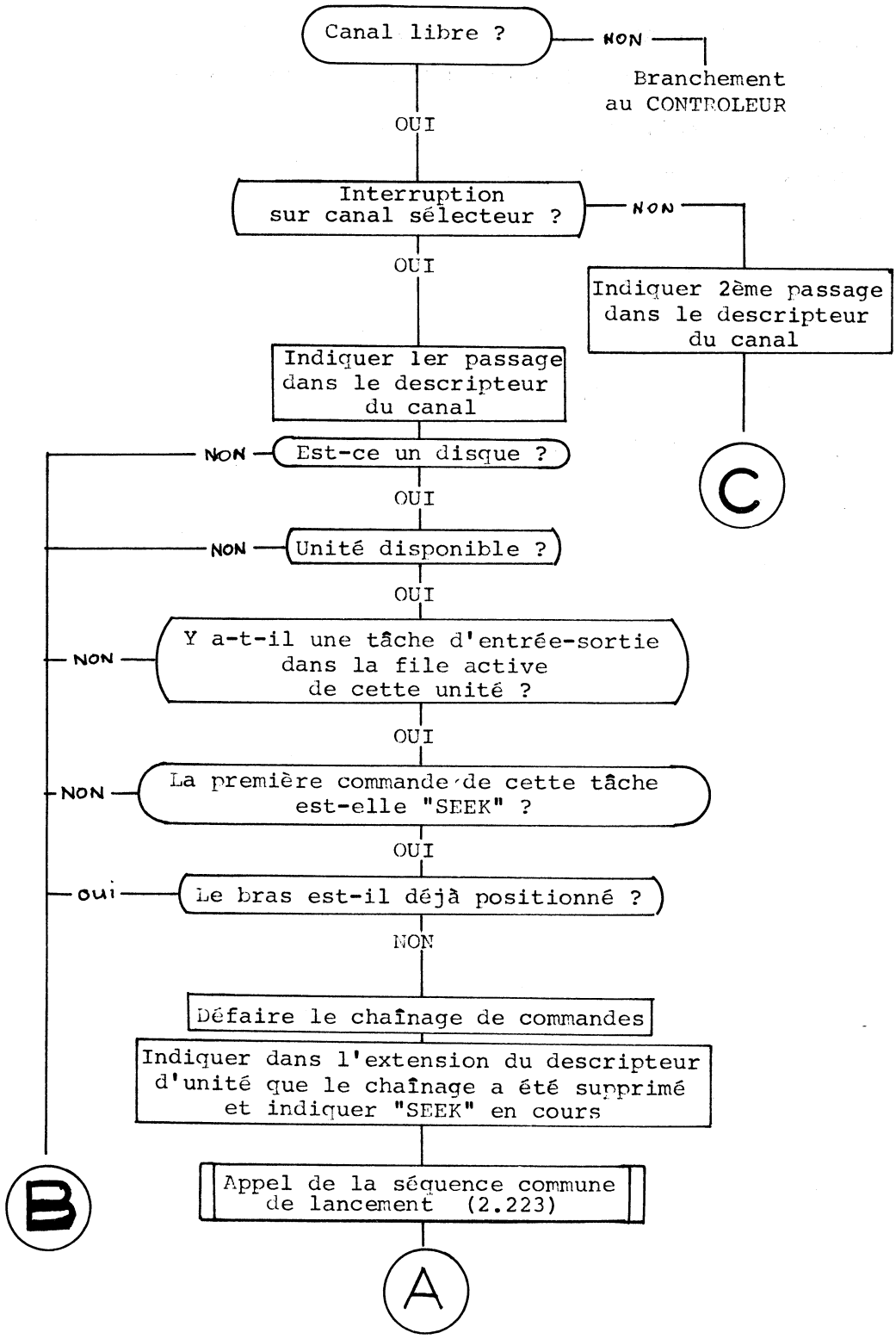
Lors du deuxième passage, le superviseur d'entrée-sortie lance une opération occupant le canal sur une des unités restant libre.

Opération de positionnement du bras

Quand il a trouvé une tâche d'entrée-sortie pour laquelle le bras doit être positionné, le superviseur d'entrée-sortie supprime le chaînage qui lie la commande SEEK à la suite du programme canal. L'initialisation de cette commande est alors faite par la séquence commune du superviseur d'entrée-sortie (2.223). Au retour de cette séquence, une boucle sur une instruction TIO permet d'attendre la libération du canal, puis de lancer une opération sur une autre unité. Quand l'interruption indiquant la fin du positionnement du bras se produit, le chaînage est rétabli et l'entrée-sortie complète initialisée.

La description de la configuration "hardware" du système, telle qu'elle est présentée en 2.21, permet de retrouver, à partir de l'unité qui a causé l'interruption, toutes les unités utilisant le même sous-canal, et ainsi de réaliser la relance. On remarque que, dans la configuration décrite en 2.21, il y a une seule unité par sous-canal sur le canal multiplex, et aucune de ces unités n'est un disque. A la suite d'une interruption provenant de ce canal, la relance se limite à l'unité qui a causé l'interruption.

La relance de l'activité canal est décrite sur la figure 2.34. A la sortie de cette séquence, on se branche au contrôleur : il activera une tâche système ou une tâche utilisateur en fonction des modifications dans les tâches apportées par l'interruption.


Canal libre ?
NON
Branchement au CONTROLEUR
OUI
Interruption sur canal sélecteur ?
NON
OUI
Indiquer 2ème passage dans le descripteur du canal
Indiquer 1er passage dans le descripteur du canal
C
NON
Est-ce un disque ?
OUI
NON
Unité disponible ?
OUI
NON
Y a-t-il une tâche d'entrée-sortie dans la file active de cette unité ?
OUI
NON
La première commande de cette tâche est-elle "SEEK" ?
OUI
oui
Le bras est-il déjà positionné ?
NON
Défaire le chaînage de commandes
Indiquer dans l'extension du descripteur d'unité que le chaînage a été supprimé et indiquer "SEEK" en cours
B
Appel de la séquence commune de lancement (2.223)
A

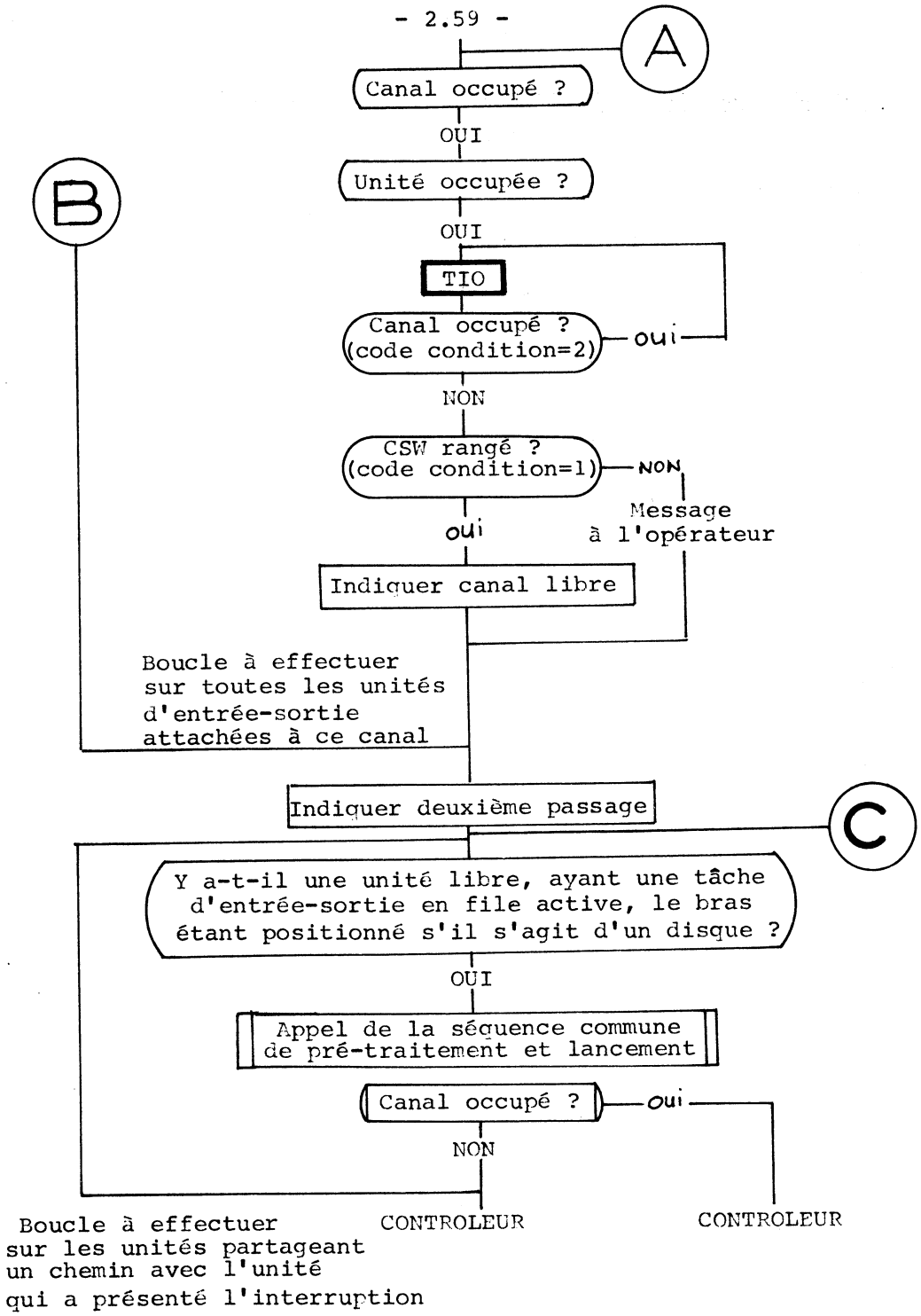


Figure 2.34 Relance de l'activité du canal

2.4 Réalisation d'un interface d'entrée-sortie CMS-GMS

Nous avons vu, au paragraphe 2.1, que GMS génère, pour plusieurs copies du système CMS, les ressources virtuelles d'entrée-sortie que celui-ci attend. Le système CMS, ou les programmes qui tournent sous son contrôle, se servent, pour effectuer des entrées-sorties, des mêmes fonctions que celles qu'ils utilisent quand le système CMS tourne en "stand-alone".

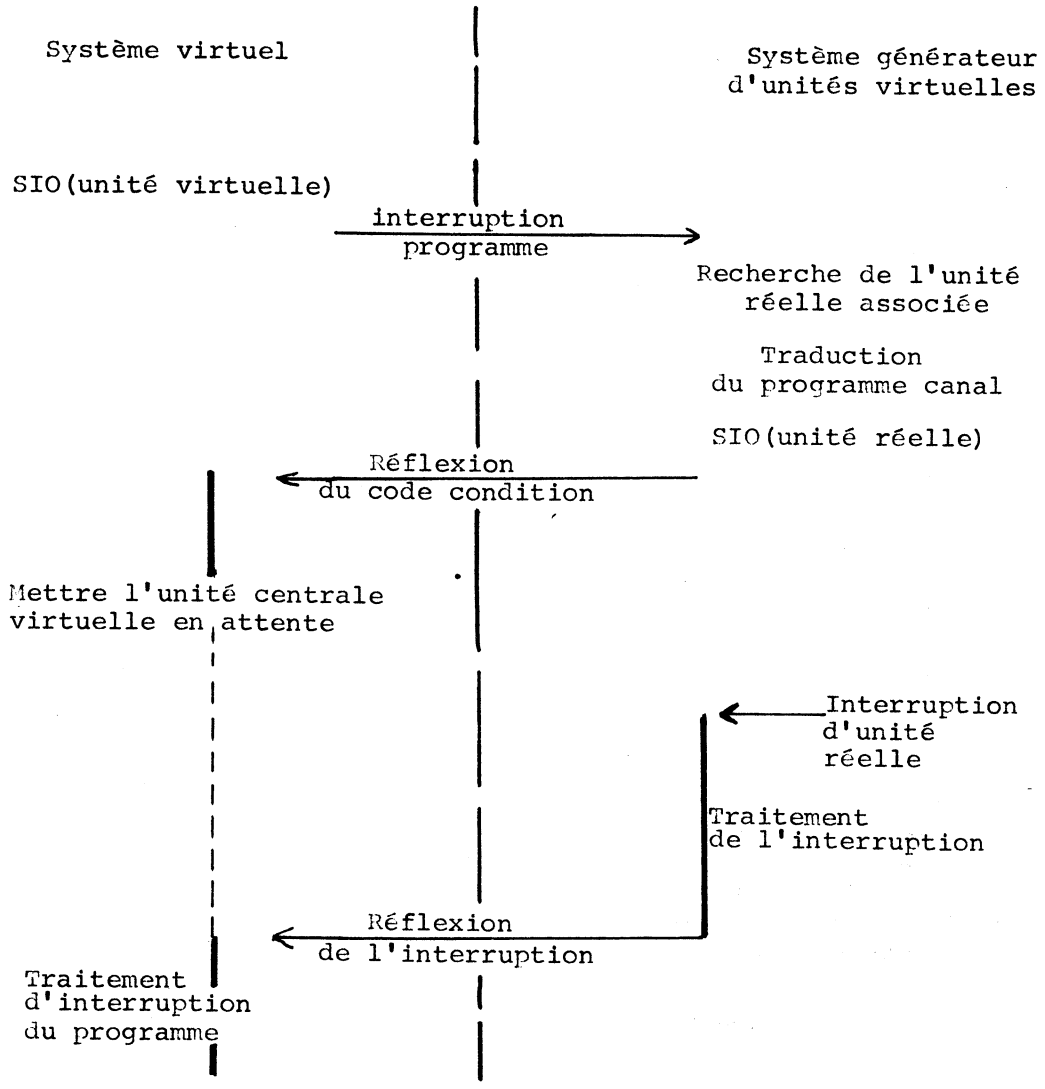
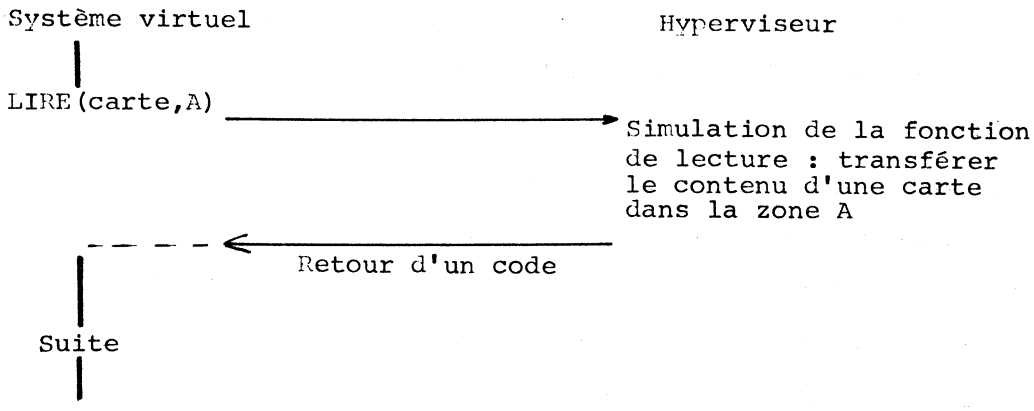


Figure 2.35 Simulation d'une entrée-sortie virtuelle

Dans un système qui génère des machines virtuelles, le programme qui est exécuté sur une machine virtuelle prépare les opérations d'entrée-sortie comme s'il disposait des unités d'entrée-sortie réelles : il construit un programme canal et le lance par une instruction SIO. Le système virtuel se déroulant en mode esclave, cette instruction privilégiée cause une passation du contrôle de l'unité centrale du système virtuel au système hyperviseur. Ce dernier transforme la demande d'entrée-sortie fournie par le sytème virtuel en une opération équivalente sur le support réel qui simule l'unité virtuelle. Ce support réel peut être une unité entièrement dédiée à un utilisateur, ou une unité partagée entre plusieurs utilisateurs, ou un fichier du système. Réciproquement, la réponse de ces supports réels doit être convertie et transmise aux utilisateurs qui ont émis les entrées-sorties. Les programmes qui font cette liaison constituent l'interface d'entrée-sortie CMS-GMS (figure 2.35). Cette simulation au niveau des unités est ainsi faite, par exemple, dans CP67 (1.2). Elle tend à diminuer l'efficacité du système, si on définit celle-ci comme étant le rapport du temps passé en mode problème sur le temps total d'activité de l'unité centrale. Aussi est-il préférable, quand c'est possible, de saisir l'accès aux ressources d'entrée-sortie dans CMS quand cet accès se fait encore au stade logique (réf BH), avant que le programme canal ne soit construit (figure 2.36). Encore faut-il que cet accès logique soit défini clairement. Dans ces conditions, une demande logique d'entrée-sortie peut être soumise directement à l'unité logique défini dans le système hyperviseur, sans qu'il n'y ait de redondance. Nous allons examiner maintenant l'interface de chaque type d'unité supportée par CMS. Il faut remarquer que la conception de ces interfaces est très dépendante de la connaissance que nous avons du système CMS (réf LF).



Simulation d'un accès logique par l'hyperviseur

Figure 2.36

Lecteur-perforateur de cartes - Imprimante

L'utilisation de ces unités sera étudiée plus complètement au paragraphe suivant. Notons toutefois que ces unités se prêtent bien à un accès logique simple, car ce sont des unités à accès entièrement séquentiel, ne transmettant des données que dans un sens unité centrale-unité pour l'imprimante et le perforateur de cartes, unité-unité centrale pour le lecteur de cartes.

Console opérateur

Le système CMS "stand-alone" communique avec un utilisateur par l'intermédiaire de la console du pupître 1052. Elle admet des commandes de lecture et d'écriture. On y dispose aussi de la touche "Attention" permettant de générer une interruption asynchrone. Un terminal 2741 admet de même des commandes de lecture et d'écriture, et sa touche "Attention" permet de générer des interruptions asynchrones, si une commande "Prepare" a été lancée auparavant (2.233).

La console virtuelle d'une copie du système CMS sera donc simulée par un terminal 2741.

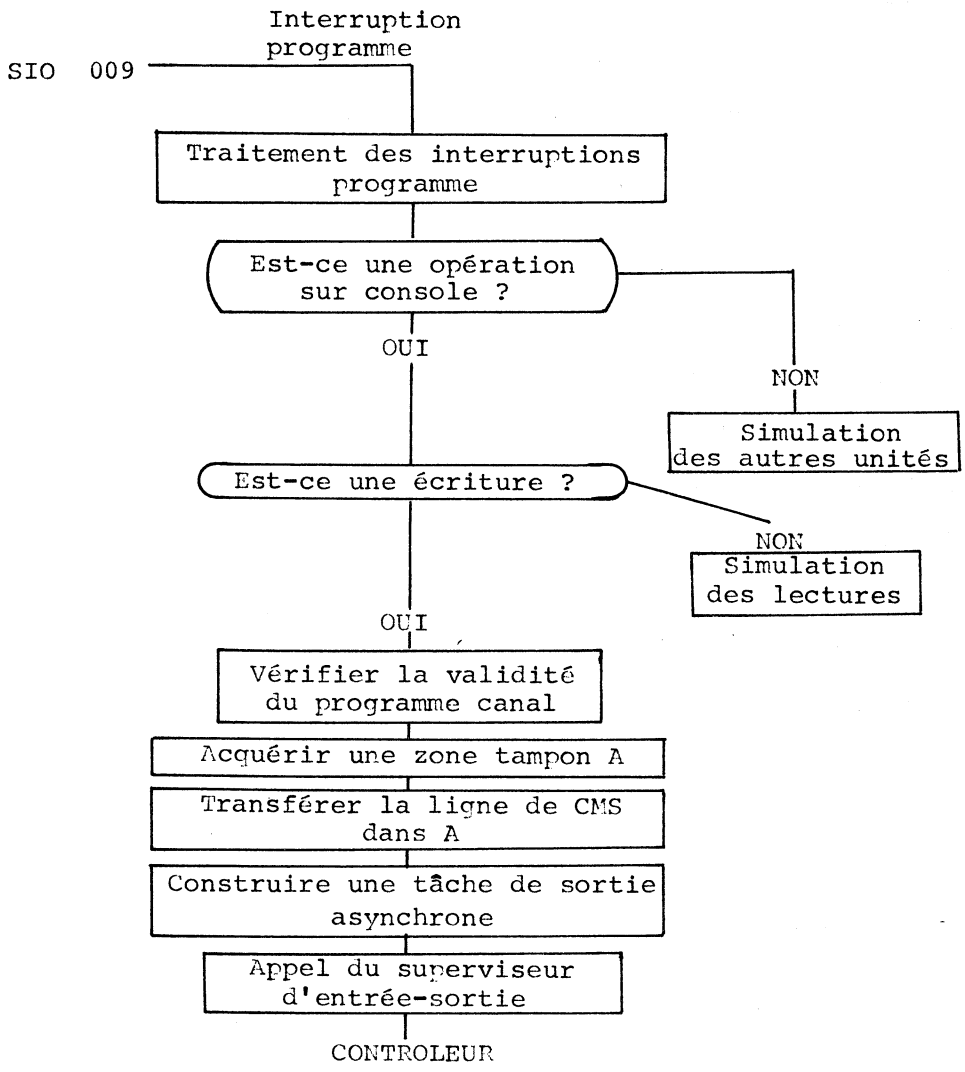
Il n'existe pas, dans CMS, un accès logique à la console par lequel passe toute demande d'entrée-sortie et qui puisse être transformé en accès réel par GMS. Les accès des programmes d'application à la console peuvent certes tous passer par la fonction WAITRD de CMS. Il serait très simple de passer le contrôle à ce moment à GMS, et de ne rendre le contrôle à CMS que lorsque l'utilisateur aurait tapé une ligne. Mais l'appel à la fonction WAITRD n'entraîne pas systématiquement une lecture sur la console. En effet, un utilisateur a la faculté de taper plusieurs lignes logiques sur la console dans une seule ligne physique. Les lignes logiques sont séparées par un caractère reconnu par CMS comme délimiteur de ligne. Ce caractère peut être modifié par l'utilisateur. Lorsqu'elle est appelée, la fonction WAITRD fournit à l'appelant une seule ligne logique. Si on laisse à GMS le soin d'exécuter l'accès logique défini par l'appel à WAITRD, GMS devra déterminer ces lignes logiques, ce qui n'est pas son rôle.

Il y a d'autre part un cas où CMS fait ses entrées directement par SIO sur la console, sans utiliser d'accès logique. C'est dans l'environnement de mise au point, qui doit être indépendant des fonctions de CMS.

Il faut donc laisser CMS transformer l'accès logique en accès physique (i.e. construire un programme canal, puis émettre une instruction SIO). GMS reçoit le contrôle lorsque l'unité centrale essaie d'exécuter cette instruction. Il entreprend alors l'analyse du programme canal, et vérifie la validité de son contenu : les commandes sont-elles valides, les adresses des données ne couvrent-elles pas une zone de GMS? Puis il construit une tâche d'entrée-sortie et fait appel au superviseur d'entrée-sortie, qui chaîne cette tâche au descripteur du terminal associé à l'utilisateur.

Dams le système CMS, les écritures sur la console se font sans attente. Cette caractéristique a été réutilisée dans GMS pour les entrées-sorties provenant de CMS. Cela permet de réactiver CMS sans attendre la fin de la sortie.

Demande d'écriture de CMS



Simulation d'une écriture sur console 1052

Figure 2.37

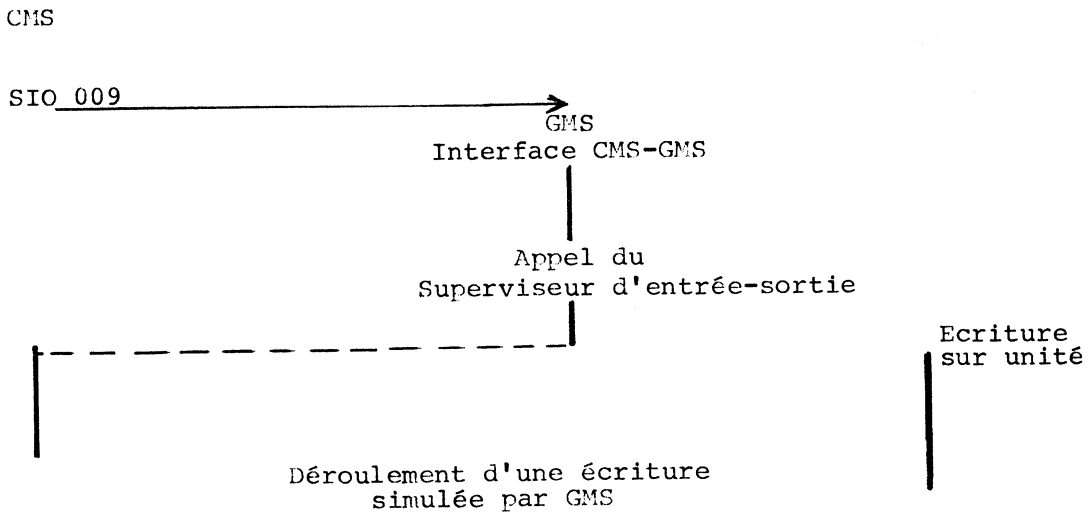
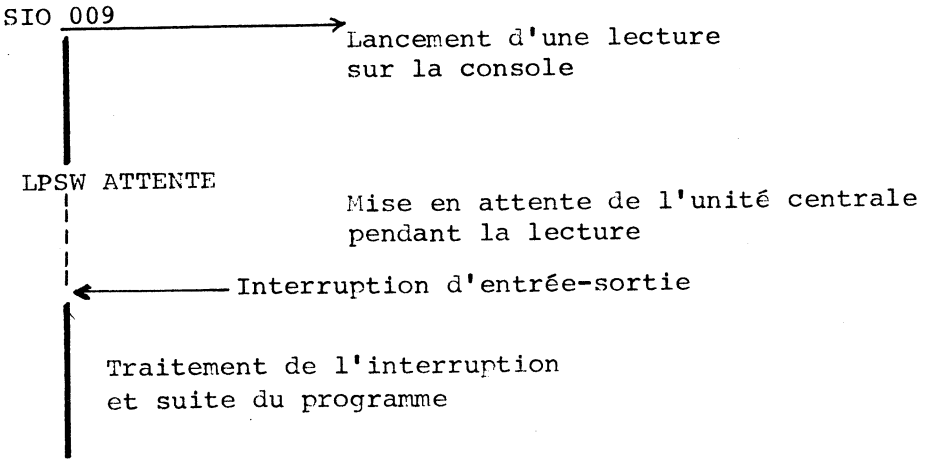


Figure 2.38

En entrée, CMS se met en attente de fin d'entrée-sortie avant d'exploiter la ligne. Le déroulement d'une lecture de ligne, en "stand-alone", est indiqué sur la figure 2.39.



Opération de lecture en CMS "stand-alone"

Figure 2.39

Pendant que des données sont transférées dans l'espace adressable d'un utilisateur, celui-ci doit être vérrouillé en mémoire centrale. Comme on ne peut le verrouiller pendant tout le temps de l'entrée à la console car il bloque alors les autres utilisateurs, on fait les démarches suivantes :

-Acquérir une zone tampon A dans la mémoire libre de GMS (figure 2.40).

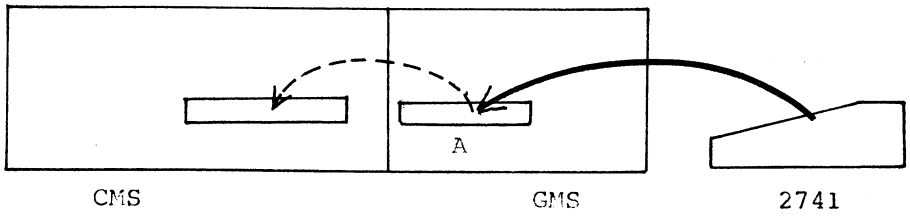
-Préparer le mot d'état canal à fournir à CMS lorsqu'on lui réfléchira l'interruption :

-adresse du dernier CCW + 8

-état "fin sur unité" + "fin sur canal"

Ce mot d'état canal est preservé dans le descripteur de la console virtuelle, rattachée au descripteur de la tâche utilisateur.

-Désactiver l'utilisateur en indiquant qu'une lecture a été demandée sur son terminal. Appeler le superviseur d'entrée-sortie en lui fournissant une tâche d'entrée-sortie synchrone demandant une lecture dans la zone tampon A.

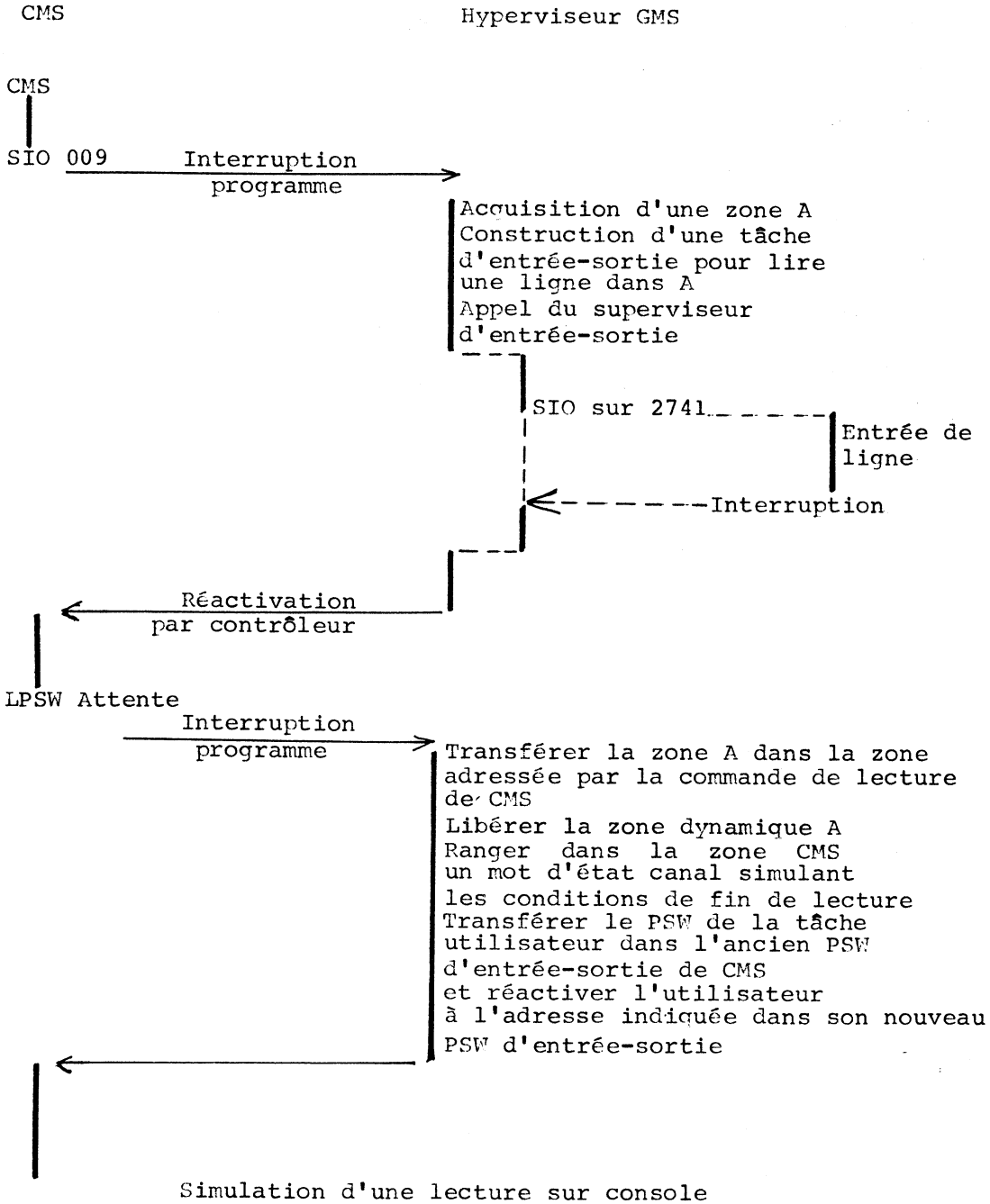


Transfert d'une ligne d'un terminal dans CMS

Figure 2.40

On ne réfléchit à CMS que des interruptions indiquant une fin correcte. En effet, on ne peut laisser à CMS le soin de corriger les erreurs du 2741, la différence de fonctionnement "hardware" entre 1052 et 2741 étant trop grande.

La tâche d'entrée-sortie étant synchrone, l'interface ne reprend le contrôle que lorsque l'entrée est terminée correctement, le traitement éventuel d'erreur ayant été fait par le superviseur d'entrée-sortie . On trouve dans le descripteur de tâche d'entrée-sortie le nombre de caractères frappés par l'utilisateur. La ligne frappée est traduite du code 2741 en code EBCDIC (code du 1052). L'adresse physique de l'unité virtuelle en cause est conservée dans la tâche utilisateur. On indique alors au contrôleur que cette tâche utilisateur peut être réactivée. Le contrôle de l'unité centrale sera donnée à cette tâche derrière l'instruction SIO, en indiquant dans le mot d'état programme un code condition nul, qui indique une initialisation correcte. CMS va émettre une instruction privilégiée LPSW, chargeant son mot d'état programme avec le bit "Attente" à 1. Cette instruction est interceptée par GMS comme instruction privilégiée. S'apercevant qu'il y a une interruption à réfléchir, grâce au contenu de la tâche utilisateur, GMS transfère dans la zone utilisateur toutes les informations utiles reflétant l'interruption : mot d'état canal, mot d'état programme. L'utilisateur peut alors être immédiatement réactivé.



Simulation d'une lecture sur console

Figure 2.41

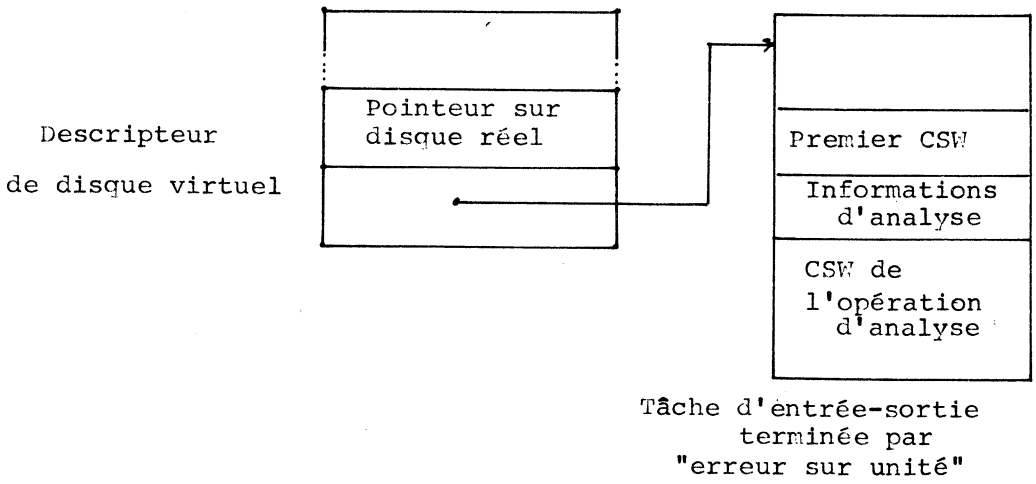
Disques

Un utilisateur de CMS a à sa disposition de manière standard un disque sur lequel se trouvent les modules du système non résidents en mémoire centrale, et un disque privé sur lequel il conserve ses fichiers. Le disque système est accessible seulement en lecture, aussi peut-on partager son accès entre tous les utilisateurs. Par contre, le contenu du disque privé peut être modifié au gré de l'utilisateur. Aussi attribue-t-on aux utlisateurs un disque réel qui est une réplique exacte du disque virtuel. On laisse CMS préparer le programme canal, et ce dernier peut être utilisé tel quel par GMS. Le seul changement que GMS doit faire est relatif à l'adresse du disque virtuel : elle est la même pour toutes les copies du système CMS, mais chaque disque réel attaché à un utilisateur a une adresse qui lui est propre. La simulation d'une entrée-sortie sur disque est la même que pour les entrées-sorties sur console, à une exception près. Nous avons dit que le traitement d'erreur diffère beaucoup entre 1052 et 2741, aussi le traitement d'erreur des 2741 est toujours fait dans le superviseur d'entrée-sortie, ou par une procédure qu'il appelle. Pour les disques, l'unité virtuelle est la réplique exacte de l'unité réelle. Nous avons laissé à CMS le soin de traiter toutes les interruptions des disques, consécutives à des opérations qu'il a demandées :

- Une tâche d'entrée-sortie sur disque a le type SCR, ainsi le traitement d'interruption par le superviseur d'entrée-sortie est réduit au minimum.
- Quand le système CMS se met en attente par LPSW, on lui réfléchit le mot d'état canal tel qu'il se trouve dans le descripteur de tâche d'entrée-sortie Si ce mot d'état canal indique "erreur sur unité", le superviseur d'entrée-sortie a automatiquement émis une

commande d'analyse, et son résultat se trouve dans le descripteur de tâche d'entrée-sortie (2.234). Le système CMS, détectant "erreur sur unité" dans le mot d'état canal, émet lui-même une commande d'analyse. Le résultat de cette commande est immédiatement mis à sa disposition par GMS à partir du descripteur de tâche d'entrée-sortie qui peut ensuite être libéré (figure 2.42). Il faut noter qu'il ne serait pas valable que le superviseur d'entrée-sortie émette une commande d'"analyse" seulement quand CMS le demande : dans ce cas, il est possible qu'une entrée-sortie pour GMS ("swapping") soit faite entre la commande qui a causé l'erreur et la commande d'analyse de CMS. Or une commande détruit les informations d'analyse dans les circuits lorsque l'erreur n'est pas permanente. L'information d'analyse ne serait alors plus valide.

Ce problème n'existe pas en CMS "stand-alone" puisque CMS, gérant lui-même ses entrées-sorties, est au courant du risque.



Description d'un disque virtuel

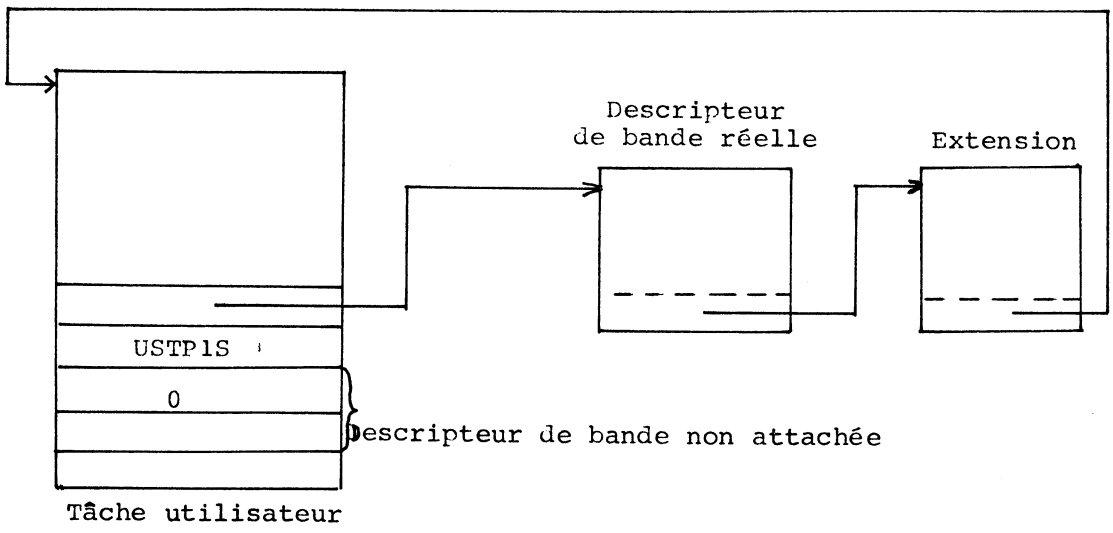
Figure 2.42

Bandes

Les bandes sont d'un emploi peu fréquent. Aussi sont-elles attachées en cours de session aux utilisateurs qui en font la demande, et leur simulation se fait de la même façon que pour les disques, par tâches SCR.

La commande d'attachement ATTACH doit être émise dans l'environnement GMS par l'utilisateur avant qu'il ne lance des opérations sur bande. Elle initialise, dans le descripteur de tâche utilisateur le pointeur, descripteur de bande virtuelle, avec l'adresse du descripteur de l'unité de bande réelle dédiée à l'utilisateur. Ce pointeur est nul tant que la commande ATTACH n'a pas été émise. Dans l'extension du desripteur de l'unité de bande, on initialise un pointeur sur le descripteur de la tâche utilisateur à laquelle est dédiée l'unité (figure 2.43). Le système CMS utilisé dans GMS a été généré avec des adresses d'unités de bande qui sont les adresses réelles des bandes dans la configuration disponible. Il n'y a donc pas de conversion d'adresse d'unité à faire.

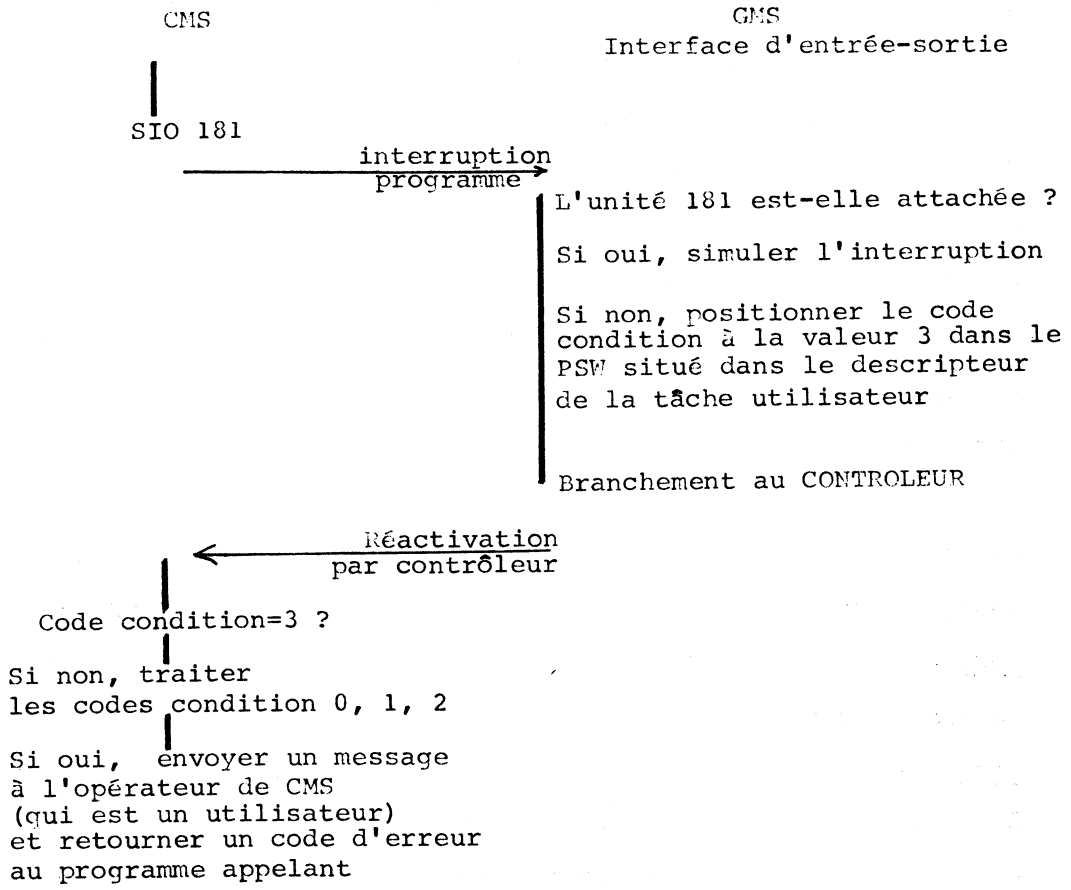
Quand CMS émet un SIO sur une bande, le contrôle de l'unité centrale est donné à l'interface d'entrée-sortie CMS-GMS. Il vérifie que l'unité virtuelle se trouve dans la configuration virtuelle de l'utilisateur. Si ce n'est pas le cas (le descripteur d'unité virtuelle contient une valeur nulle), on réfléchit immédiatement à CMS un code condition 3, simulant ainsi une unité non opérationnelle. CMS réagit comme s'il se trouvait en "stand-alone", en prévenant l'utilisateur : il n'a donc pas besoin d'être modifié (figure 2.44). Si l'unité virtuelle est bien attachée, le traitement de l'entrée-sortie dans GMS se déroule comme pour les disques.



USTP1S : pointeur sur une tâche d'entrée-sortie contenant des informations d'analyse

Description des bandes virtuelles

Figure 2.43



Opération d'entrée-sortie
sur une bande non attachée

Figure 2.44

2.5 Exemple : entrées-sorties sur imprimante

Les entrées-sorties sur imprimante, depuis l'accès logique demandé par CMS jusqu'à la sortie des lignes sur l'imprimante réelle, sont étudiées dans ce chapître. Comme il y a une seule imprimante dans la configuration, on ne peut la laisser en accès direct à un ou plusieurs utilisateurs. Nous avons donc utilisé la technique bien connue de "spooling". Cette technique consiste à associer à chaque utilisateur qui se sert d'une imprimante une unité logique enregistrant les données sur un support à accès direct. Les utilisateurs peuvent alors travailler en simultanéité, du point de vue d'un observateur extérieur, sur les imprimantes logiques.

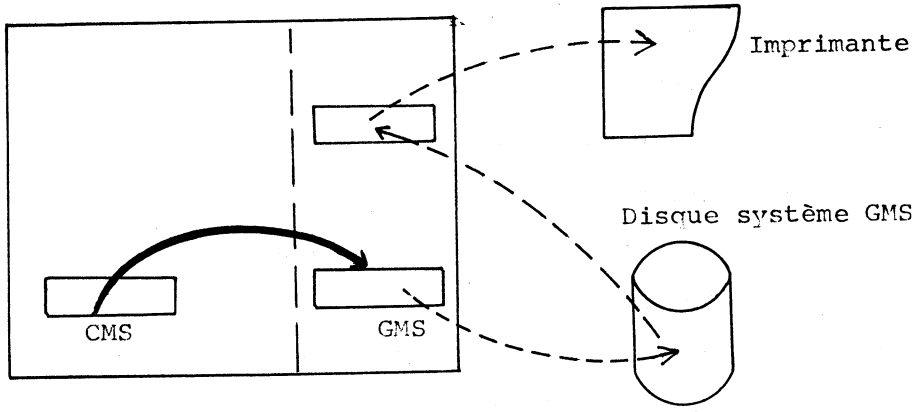
GMS dispose d'une gestion de fichiers lui permettant d'accéder et de ranger des informations sur le disque système. Un fichier est désigné par un nom composé de trois éléments : nom1.nom2.nom3.
nom1 désigne le destinataire du fichier : ce peut être une unité physique, par exemple l'imprimante, ou un utilisateur du système.
nom2 désigne le créateur du fichier, par exemple le nom de l'utilisateur qui a créé un fichier destiné à l'imprimante.
nom3 représente, pour tous les fichiers de "spooling", un numéro de série. Pour les autres fichiers, sa signification est laissée à l'appréciation du créateur du fichier.

A un instant donné, l'imprimante virtuelle d'un utilisateur est associée à un fichier de GMS.

Un ensemble de lignes envoyé par un utilisateur de CMS pour être sorti sur imprimante est manipulé en deux temps par GMS (figure 2.45) :

1- Transfert des lignes sur le fichier du disque système associé à l'imprimante virtuelle de l'utilisateur : ces transferts se font en synchronisation avec les demandes émises par les utilisateurs.

2- Quand l'utilisateur a sorti un ensemble cohérent d'informations, par exemple la liste d'un assemblage, il l'indique à GMS, qui peut alors transférer le contenu du fichier sur l'imprimante.



Opérations de "spooling" sur imprimante

Figure 2.45

2.51 Imprimante virtuelle

L'accès du système CMS ou d'un programme d'application tournant sous ce système se fait d'abord par un accès logique. Cet accès logique est l'appel à la fonction PRINTR de CMS, en lui fournissant dans le registre 1 l'adresse d'une liste de paramètres. En assembleur 360, cette liste a le format suivant :

```
DC    CL8'PRINTR'  Nom de la fonction appelée
DC    AL4(data)    Adresse de la ligne à imprimer
DC    F'n'         Nombre d'octets à transférer
```

Le premier octet qui se trouve en tête des données à transférer est un caractère de contrôle : il n'est pas imprimé mais il est interpreté par la fonction PRINTR pour contrôler l'avancement du papier sur l'imprimante : sauter une ligne, deux lignes, sauter à la page suivante, etc... PRINTR transforme alors la demande qu'il a reçue en une opération d'entrée-sortie qu'il fait exécuter par l'imprimante.

-Il vérifie la validité de la longueur des données : cette longueur ne doit pas dépasser le maximum d'une ligne sur l'imprimante (132 caractères) ni être négative ou nulle.

-Il interprète le caractère de contrôle et construit un commande canal effectuant le saut désiré.

-Il construit un mot de commande canal chaîné au précédent et effectuant l'impression de la ligne.

-Il exécute un SIO

-Il attend la fin de l'opération et traite les erreurs éventuelles

-Il retourne à l'appelant un code indiquant comment s'est déroulée l'opération.

Tous les accès de CMS à l'imprimante se font par appel à la fonction PRINTR par une instruction SVC (Supervisor Call). cette instruction crée une interruption ; toutes les informations nécessaires à l'exécution de la fonction PRINTR sont fournis en paramètres ou sont adressées par un de ces paramêtres. Cela permet de supprimer dans CMS le travail fait par PRINTR, et de laisser GMS accomplir l'accès logique réalisé par cette fonction. GMS fait la vérification de la validité des paramêtres tel que CMS la faisait, transfère dans l'imprimante logique les données si elles sont valides, et réfléchit à CMS le même code que celui retourné par PRINTR (figure 2.46).

Lors de la première écriture sur l'imprimante, GMS "crée" une imprimante logique : il demande la création d'un fichier de "spooling" au nom de l'utilisateur et obtient ainsi un numéro de fichier ; il met dans le descripteur de la tâche utilisateur l'adresse du descripteur d'imprimante virtuelle. Les enregistrements physiques des fichiers sur le disque système ont une taille de 496 octets. On peut donc mettre dans un enregistrement physique plusieurs lignes d'imprimante. Cela nécessite cependant la réservation d'une zone de 496 octets dans la zone de mémoire centrale de GMS, ceci pour chaque utilisateur possédant une imprimante logique. L'adresse de cette zone, ainsi que le descripteur du fichier de "spooling", sont contenus dans le descripteur d'imprimante virtuelle.

On ne peut lancer l'impression d'un fichier de "spooling" dès qu'il contient au moins un enregistrement, car il y aurait sur une seule page des lignes appartenant à plusieurs utilisateurs. L'impression d'un tel fichier n'est faite que lorsque l'utilisateur, via la fonction CLOSIO de CMS, indique que son fichier peut être imprimé. Il est à noter que cette fonction n'a pas d'utilité dans CMS "stand-alone" puisque les lignes sortent directement sur l'imprimante qui n'est pas partagée.

Remarque : Dans un souci d'optimisation, tous les caractères blancs situés à l'extrême droite de la ligne ne sont pas enregistrés sur disque. Ceci permet d'augmenter le nombre de lignes rangées dans un bloc physique de 496 octets, et de diminuer ainsi le nombre d'entrées-sorties sur disque et le temps d'occupation de l'imprimante.

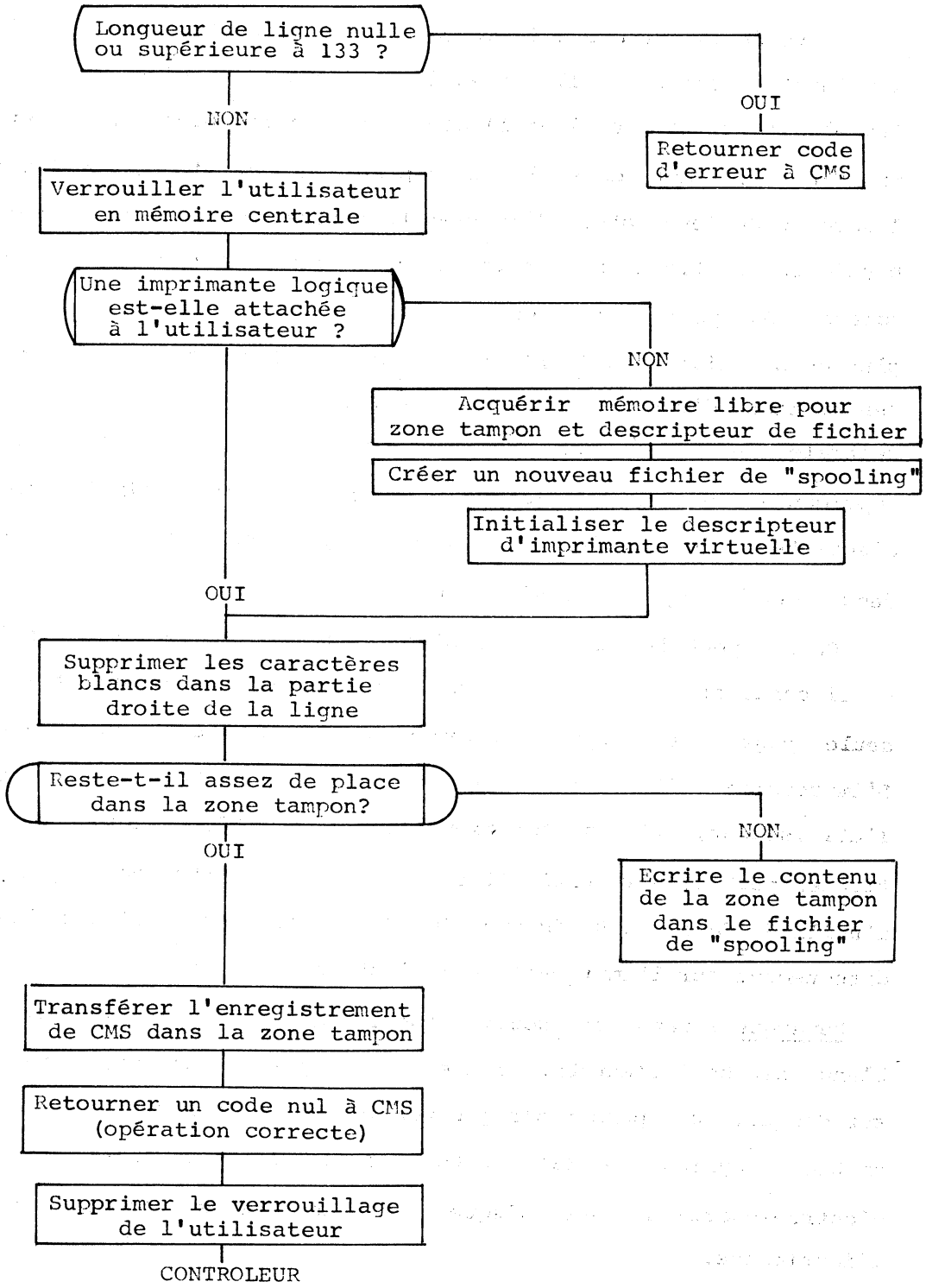


Figure 2.46 Sortie sur imprimante virtuelle

2.52 Transfert du fichier de "spooling" sur l'imprimante

Quand un utilisateur indique qu'il libère une imprimante logique, GMS active une tâche système chargée de transférer sur l'imprimante les fichiers qui lui sont destinés. Ces fichiers sont imprimés dans l'ordre croissant des numéros de série (nom3), c'est-à-dire par ordre d'ancienneté. L'imprimante étant une unité très lente relativement à la vitesse de transfert d'un disque, on utilise la technique des zones alternées. Elle permet de faire travailler l'imprimante à vitesse maximum, sans temps mort dû à l'attente d'une lecture sur le disque système.

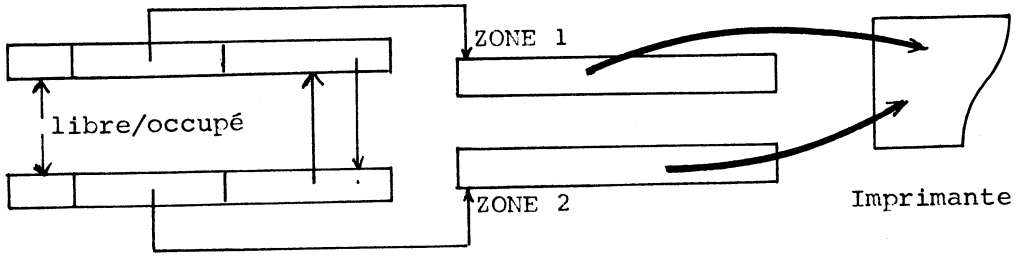


Figure 2.47 Utilisation des zones alternées pour le "spooling"

La tâche système demande la lecture synchrone d'un enregistrement physique du fichier dans A. Quand cette lecture est achevée, on indique dans A qu'il contient des lignes à imprimer, et on appelle une procédure qui crée une tâche asynchrone d'entrée-sortie pour imprimer le contenu de A. Puis on passe au bloc suivant (en l'occurence B). S'il est libre, on demande la lecture d'un nouvel enregistrement dans B, on indique B occupé, et on appelle à nouveau la procédure d'impression sans attente de E Si le bloc B est occupé, (son contenu n'a pas encore été imprimé), cette tâche système est mise en attente. C'est la procédure de traitement de fin d'entrée-sortie RTND (2.234) qui indique, quand l'impression est terminée, que ce bloc est libre et qui réveille la tâche de "spooling" (figure 2.49).

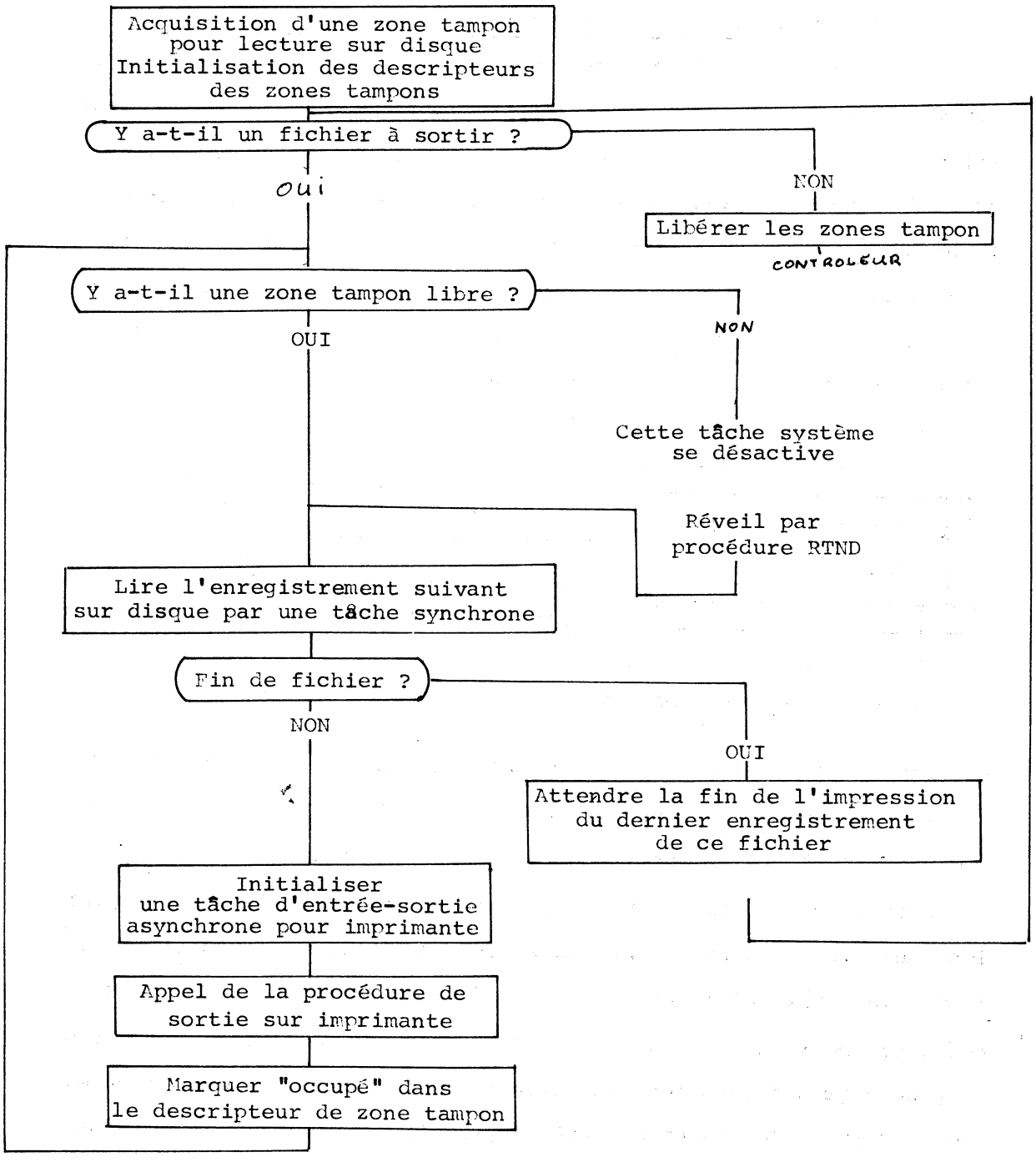
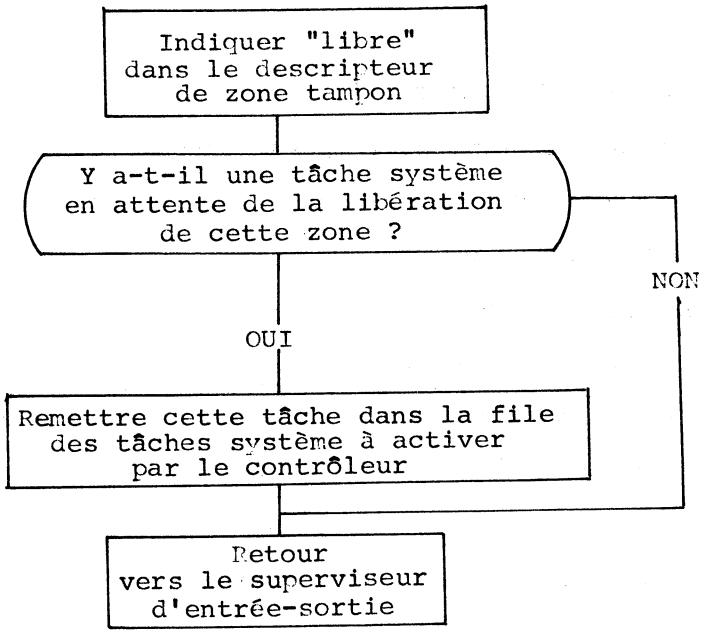


Figure 2.48 Sortie des fichiers de "spooling" sur imprimante



Procédure RTND d'imprimante employée
par la tâche de "spooling"

Figure 2.49

Un enregistrement contenant un nombre entier de lignes, une seule tâche d'entrée-sortie sur imprimante peut faire imprimer toutes les lignes contenues dans le bloc, en utilisant le chaînage de commandes ; elle ajoute si nécessaire des commandes pour remplir les fonctions indiquées par les octets de contrôle.

Conclusion

Le système GMS dans lequel est inclus ce superviseur d'entrée-sortie est maintenant opérationnel. Pour le réaliser, nous avons dû concevoir des mécanismes suffisamment généraux pour qu'il puisse tenir compte du système CMS. Les mécanismes de tâches asynchrones et de tâches auto-correctrices se sont révélés particulièrement bien adaptés aux situations où nous les avons employés. Les algorithmes développés ici pourraient être utilisés avec profit pour de nombreux systèmes tournant sur des matériels équivalents ; un pas important serait franchi si cela pouvait conduire à une normalisation des entrées-sorties dans les systèmes d'exploitation.

CHAPITRE 3

QUELQUES AUTRES ASPECTS DES ENTREES-SORTIES

Nous allons aborder dans ce chapître quelques aspects complémentaires des entrées-sorties. Ces aspects n'ont pu l'être au chapître précédent car leur implantation dans GMS était impossible ou ne se justifiait pas étant donné la configuration disponible.

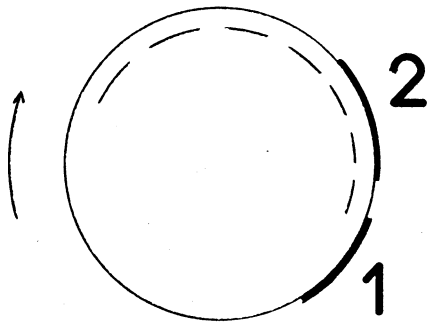
3.1 Optimisation

L'optimisation représente un aspect fondamental de l'utilisation des ordinateurs, à cause de la différence importante entre la vitesse de calcul des unités centrales et le temps d'accès aux informations sur les unités périphériques.

3.11 Optimisation par enchaînement des commandes

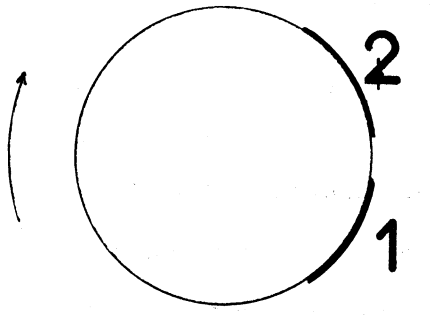
L'optimisation par enchaînement des commandes consiste à faire exécuter par un canal une suite de commandes à partir d'un seul ordre fourni à ce canal. Une condition d'interruption unique est présentée par le canal lorsque toutes les commandes de la chaîne ont été exécutées. On économise ainsi le temps d'unité centrale qui aurait été nécessaire au lancement individuel de ces commandes et au traitement des interruptions qui indiquent la fin de leur exécution. Le chaînage de commandes a été employé dans GMS (2.52) dans les opérations de "spooling" pour faire écrire plusieurs lignes par une seule tâche

d'entrée-sortie. Ce chaînage peut aussi être utilisé pour optimiser l'utilisation des unités à accès sélectif, pour accéder à plusieurs enregistrements consécutifs (figure 3.1). En plus du gain de temps d'unité centrale, on peut économiser le temps de plusieurs rotations de l'unité, si elle est mobile : quand il y a enchaînement des commandes, après accès au premier enregistrement, le canal a le temps de lire la commande suivante avant que la tête de lecture-écriture ne passe sur l'enregistrement qui suit. Ce n'est généralement pas possible s'il faut que l'unité centrale accepte et traite une interruption entre deux commandes.



Lecture sans chaînage de commande

Premier tour : Lecture de l'enregistrement 1
et traitement d'interruption
par l'unité centrale (en pointillé sur la figure)
Deuxième tour : Lecture de l'enregistrement 2



Lecture avec chaînage de commandes

Premier tour : lecture du premier enregistrement,
puis lecture par le canal
de la deuxième commande en mémoire centrale,
puis lecture du deuxième enregistrement

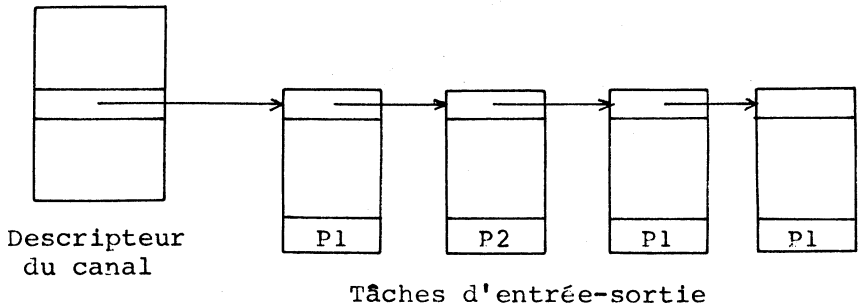
Figure 3.1

Cette optimisation peut être faite par le module qui gère ces unités (c'est le superviseur d'entrée-sortie quand il y en a un dans le système considéré) ; elle peut aussi être faite par le programme qui appelle ce module. La première solution semble la plus logique, étant donné le rôle d'optimisation attribué au superviseur d'entrée-sortie. Cependant, la deuxième solution est parfois plus efficace. C'est ce qui se passe par exemple pour l'imprimante : aucune autre optimisation n'étant possible, le programme appelant le superviseur d'entrée-sortie lui fournit une seule tâche d'entrée-sortie : le programme canal associé à cette tâche imprime plusieurs lignes. Cela évite de créer une tâche d'entrée-sortie par ligne, on gagne ainsi la place occupée par ces tâches en mémoire centrale. Par contre, pour les unités à accès sélectif, la modification de l'ordre d'exécution des tâches pouvant être source d'optimisation, une tâche d'entrée-sortie se rapporte généralement à l'accès à un seul enregistrement (3.12).

Un problème se pose lorsque survient une erreur permanente pour une commande appartenant à une suite de commandes. Si cette suite de commandes correspond à une seule tâche d'entrée-sortie émise par un programme, l'erreur peut être transmise à ce programme. Mais si cette suite de commandes a été créée par le superviseur d'entrée-sortie, à partir de plusieurs tâches d'entrée-sortie, c'est lui qui doit déterminer à quelle tâche se rapporte l'erreur, informer le processus qui a émis cette tâche, et recréer une suite de commandes à partir des tâches indépendantes de ce processus et non encore exécutées.

3.12 Optimisation par réorganisation des demandes d'entrée-sortie

Sur les unités à accès sélectif, il est possible de réorganiser les demandes d'entrée-sortie, puisque l'accès à un enregistrement physique est du point de vue matériel indépendant des accès précédents. Cette réorganisation ne doit pas modifier l'ordre d'exécution des tâches d'entrée-sortie émises par un même processus. Pour cela, il faut pouvoir retrouver, dans une suite de tâches, celles qui sont associées à un processus donné. On peut par exemple mettre dans chaque tâche d'entrée-sortie une identification du processus émetteur. L'identification de tâches appartenant à un même processus est aussi utilisable pour les retirer de la file d'un canal lorsqu'une erreur s'est produite pour l'une d'entre elles.

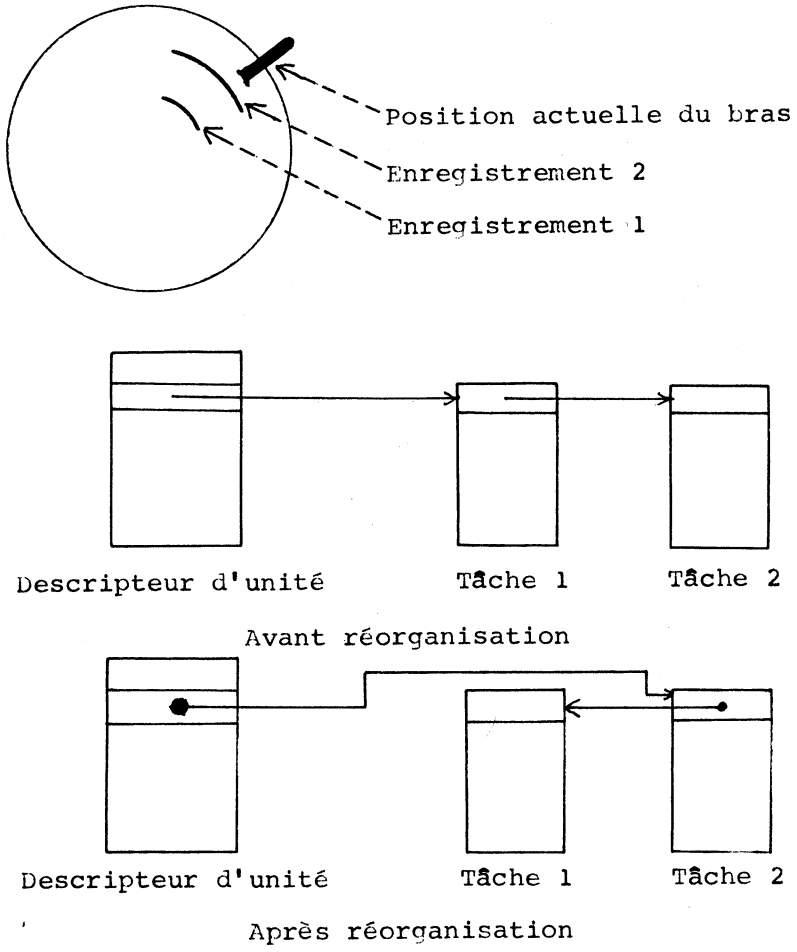


Identification des processus dans les files d'attente
de tâches d'entrée-sortie

Figure 3.2

Nous avons déjà vu au chapître précédent comment l'utilisation des unités de disque pouvait être améliorée en lançant des commandes isolées de positionnement de bras sur plusieurs unités rattachées au même canal. L'utilisation d'une unité de disque peut encore être améliorée si on cherche à minimiser le déplacement du bras entre deux accès. Pour cela, le superviseur d'entrée-sortie met en tête de file

et lance sur l'unité la tâche d'entrée-sortie pour laquelle la position du bras est la plus proche de sa position actuelle. Par exemple, dans la situation indiquée par la figure 3.3, l'enregistrement 1 sera lu après l'enregistrement 2, bien qu'il ait été soumis avant ce dernier au superviseur d'entrée-sortie.



Optimisation du déplacement du bras sur disque

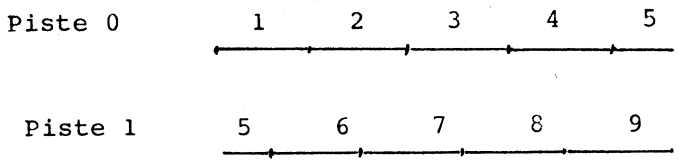
Figure 3.3

Toutefois, pour ne pas bloquer des tâches relatives à un enregistrement situé dans une zone de cylindres très peu employée, les tâches peuvent être ordonnées selon la technique de l'ascenseur : le bras est déplacé dans un sens unique, par exemple de l'intérieur vers l'extérieur. Lorsqu'une unité devient libre, les prochaines tâches exécutées sont celles relatives au cylindre sur lequel le bras est positionné, ou, s'il n'y a plus de tâche pour ce cylindre, celles relatives au cylindre le plus proche, accessible par déplacement du bras dans le sens autorisé, et pour lequel une demande d'entrée-sortie existe. Lorsque, dans ce sens de déplacement, il n'y a plus de cylindre pour lequel une tâche d'entrée-sortie existe, le sens de déplacement est inversé et l'exploration des files d'attente recommence .
Cependant, si des demandes trop rapprochées sont faites pour le cylindre sur lequel le bras est positionné, les autres tâches peuvent être très retardées : on peut y remédier en ne lançant que les tâches qui se trouvaient dans la file du cylindre lorsque ce dernier a été atteint.

Une autre optimisation des unités à accès sélectif peut être faite par réorganisation des tâches au niveau d'un cylindre pour les disques à têtes mobiles ou au niveau de toutes les pistes pour les unités à têtes fixes. Cette optimisation s'applique surtout aux disques à têtes fixes et aux tambours qui peuvent contenir un nombre beaucoup plus important d'enregistrements qu'un cylindre de disque à têtes mobiles (un cylindre de disque IBM 3330 contient environ 57 enregistrements de 4096 octets, alors qu'un tambour IBM 2301 en contient plus de 900). Dans ce qui suit, ce que nous dirons des tambours se rapporte aussi aux disques à têtes fixes, ces deux types d'unités ayant des

performances et des capacités voisines. Naturellement, l'ordre logique des demandes d'entrée-sortie doit toujours être respecté : les tâches d'entrée-sortie relatives à un même processus ne doivent pas être permutées dans les files d'attente.

Une piste ne contient pas toujours un nombre entier d'enregistrements. Dans ce cas, le dernier enregistrement chevauche la fin d'une piste et le début de la piste suivante (figure 3.5).



Exemple d'organisation des enregistrements sur deux pistes consécutives d'un tambour

Figure 3.5

Nous appelons piste logique un ensemble minimum de pistes adjacentes contenant un nombre entier d'enregistrements. L'origine de toutes les pistes logiques se trouve au même instant en face des têtes de lecture-écriture.

Soit s le nombre d'enregistrements par piste logique. Nous dirons que chaque enregistrement est placé dans un secteur du tambour. Un secteur regroupe tous les enregistrements d'une unité ayant le même numéro dans la piste logique à laquelle ils appartiennent. Le numéro d'ordre des enregistrements d'un secteur sur les pistes logiques est attribué au secteur (figure 3.5). Le temps d'accès à un enregistrement d'un secteur est indépendant de la piste logique à laquelle il appartient : dans un même secteur, deux enregistrements diffèrent seulement par le numéro des têtes de lecture auxquelles ils correspondent, et la commutation d'une tête à une autre ne crée pas de délai supplémentaire.

Deux algorithmes parmi d'autres peuvent être employés pour le choix

des tâches à lancer sur une unité (réf PJD, EY). On se restreint au cas où les pistes de l'unité contiennent un nombre entier d'enregistrements. Le chaînage de commandes peut être utilisé s'il existe.

Le premier algorithme consiste à exécuter les commandes dans leur ordre d'arrivée. Si les demandes d'entrée-sortie sont réparties de manière uniforme pour tous les secteurs, le temps moyen nécessaire au transfert d'un enregistrement est indépendant du nombre de demandes émises pour l'unité. Soit L la charge de l'unité, c'est-à-dire le nombre de demandes en attente. Nous supposons que le système est en équilibre, la charge est alors constante. Soit ts le temps de transfert d'un enregistrement. Le temps moyen d'exécution d'une demande est égale au temps mis par la tête de lecture-écriture pour accéder au début de l'enregistrement (délai rotationnel), augmenté du temps de transfert de l'enregistrement. Sa valeur moyenne, lorsque cet algorithme est employé, est

$$ts(1 + (s-1)/2) = ts\ (s+1)/2$$

s'il y a chaînage des demandes, et

$$ts\ (1 + s/2) = ts\ (s+2)/2$$

s'il n'y a pas chaînage des demandes.

Le deuxième algorithme consiste à définir autant de files d'attente qu'il y a de secteurs. Les files sont classées dans l'ordre croissant des numéros de secteur, et la première file est considérée comme suivant la dernière file (figure 3.6). Une tâche d'entrée-sortie est mise en queue dans la file correspondant au secteur concerné par cette tâche. Lorsque l'unité devient libre, le superviseur d'entrée-sortie établit une chaîne de commandes correspondant aux tâches prises dans l'ordre suivant : première tâche de la première file non vide i, première tâche de la première file non vide j qui suit la file i, etc... On peut arrêter le chaînage quand une boucle d'exploration est

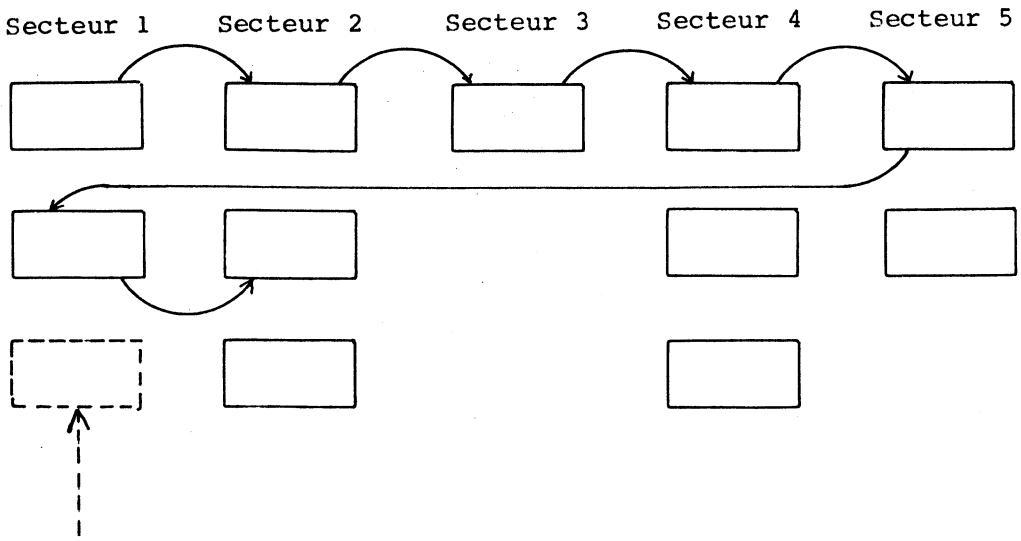
terminée, ou continuer jusqu'à épuisement des files d'attente. Le temps de traitement moyen d'une demande est

ts (1 + (s-1)/(L+1) = ts ((L+s)/(L+1))

à l'exception de la première demande de la chaîne pour laquelle le temps moyen est

ts (1 + s/(L+1)) (réf PJD)

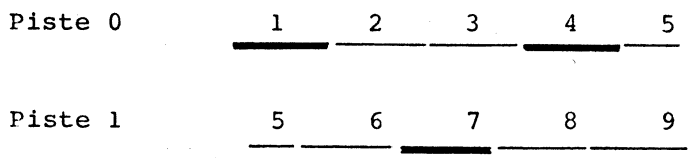
Le temps de traitement d'une demande par l'unité diminue quand la charge augmente, et cet algorithme apparaît immédiatement plus efficace que le précédent. Un tambour utilisé selon cette technique est appelé tambour de pagination.



Organisation des demandes
sur un tambour de pagination

Figure 3.6

Ces algorithmes peuvent être modifiés dans certaines situations.
Si les pistes logiques sont formées de plusieurs pistes physiques, et si les enregistrements ne se recouvrent pas, c'est-à-dire qu'ils n'ont aucune position rotationnelle commune, il est plus efficace de choisir les tâches dans l'ordre croissant des distances des enregistrements à l'origine des pistes.



Exemple d'enregistrements sans recouvrement

Figure 3.7

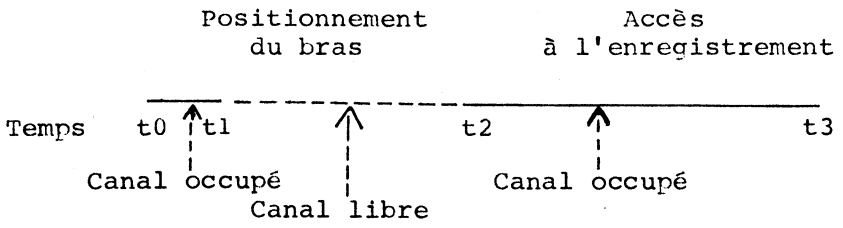
Dans le cas représenté par la figure 3.7, les enregistrements 1, 4, 7 ne se recouvrent pas. Il est préférable d'y accéder dans l'ordre 1, 7, 4 plutôt que dans l'ordre 1, 4, 7.
Une autre amélioration peut être apportée aux algorithmes de base lorsqu'il est possible de connaître la position des têtes de lecture-écriture à un instant donné. Il est alors possible de déterminer dans quelle file d'attente il faut choisir le premier élément de la suite des commandes pour que le premier accès soit effectué le plus rapidement possible. Si les enregistrements sont répartis uniformément sur l'unité, on gagne en moyenne le temps d'une demi-révolution pour le premier accès.

3.2 Les entrées-sorties dans les ordinateurs 370

Les ordinateurs IBM 370 offrent des possibilités nouvelles par rapport à la série 360 au point de vue des entrées-sorties. La compatibilité entre la série 360 (à l'exclusion du modèle 67) et la série 370 est totale dans le sens 360-370. Le superviseur d'entrée-sortie réalisé pour GMS peut être employé sans problème pour gérer des entrées-sorties sur un ordinateur 370. Moyennant quelques modifications mineures, il peut exploiter les nouveaux dispositifs pour améliorer l'efficacité du système.

3.21 Canal multiplex par bloc

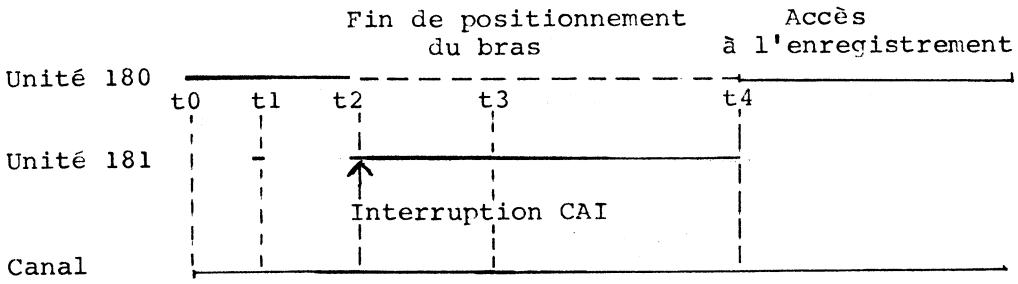
Dans les systèmes 360, un canal peut être du type sélecteur ou multiplex. Les unités rapides, telles que les tambours, ne peuvent pas être connectées à un canal multiplex qui n'a pas une vitesse de transfert suffisante. Ces unités sont donc connectées à un canal sélecteur. Mais un tel canal est inutilisable pour d'autres unités pendant l'exécution d'un programme canal sur une unité. C'est ce qui conduit à lancer isolément des commandes isolées de positionnement de bras sur un disque (2.235). Un canal multiplex par bloc a plusieurs sous-canaux et force l'unité à faire les transferts de blocs de données en mode continu. Le multiplexage est possible entre plusieurs blocs de données quand le chaînage de commandes est spécifié. Ainsi, si une commande mobilise une unité plus longuement que le canal desservant cette unité, le canal peut exécuter une commande pour une autre unité dès qu'il est libre, et exécuter ultérieurement et automatiquement la commande suivante pour la première unité. C'est ce qui se passe pour une commande positionnement de bras sur un disque (figure 3.8).



Occupation du canal pour une lecture
ou une écriture sur disque

Figure 3.8

Soit 180 l'adresse de l'unité sur laquelle se déroule l'opération indiquée sur la figure 3.8, et 181 l'adresse d'une autre unité desservie par le même canal. Si une demande d'entrée-sortie pour l'unité 181 est faite entre l'instant t1 et l'instant t2, la commande est acceptée puisque le canal est libre ; le multiplexage sera fait automatiquement par le canal. Si la demande est émise entre l'instant t0 et l'instant t1, la demande est rejetée avec l'indication "canal occupé". Il serait peu efficace de boucler sur l'instruction SIO, car l'occupation du canal dure pendant tout le transfert du bloc de données. La demande est donc mise en attente. Elle sera réactivée sur interruption. Une nouvelle interruption a été créée dans le système 370 : c'est l'interruption CAI (Channel Available Interruption). Elle informe l'unité centrale qu'un canal ayant précédemment indiqué qu'il était occupé vient d'être libéré. Ce type d'interruption est indiqué par un mot d'état canal entièrement nul.
Pour l'opération indiquée sur la figure 3.9, cette interruption se produirait à l'instant t2.

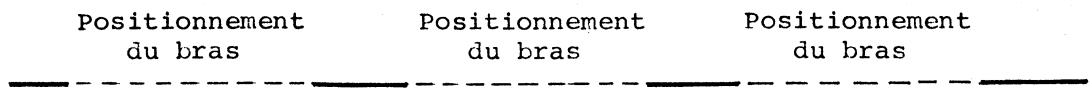


Opérations sur multiplex à bloc

Figure 3.9

Quelques modifications doivent être apportées au superviseur d'entrée-sortie pour qu'il puisse exploiter efficacement ce nouveau type de canal.

Lorsque qu'une opération d'entrée-sortie est acceptée par un tel canal (instant t0), le canal n'est pas marqué "occupé". Si une autre opération est rejetée parce que le canal répond "occupé" (instant t1), le canal doit être marqué "occupé" dans son descripteur, et la tâche d'entrée-sortie qui a demandé l'opération est mise en file d'attente. Le superviseur d'entrée-sortie, sachant désormais que le canal est occupé, mettra directement les prochaines tâches destinées à ce canal en file d'attente. Quand une interruption CAI (instant t2) ou une interruption marquant la libération du canal surviendra, l'indicateur "occupé" du descripteur du canal sera remis à zéro, et les tâches en attente de libération du canal seront initialisées, jusqu'à ce que le canal réponde à nouveau "occupé". Le canal peut ainsi être utilisé efficacement sans analyser les programmes canaux et sans lancer de commande isolée (par exemple SEEK). Ce nouveau type de canal présente beaucoup d'intérêt lorsqu'il y a dans un programme canal plusieurs commandes pour lesquelles le canal travaille pendant un temps moins long que l'unité (par exemple pour accéder à plusieurs enregistrements situés sur des cylindres différents d'un disque).



Trait continu : canal occupé par l'unité
Trait discontinu : canal libre

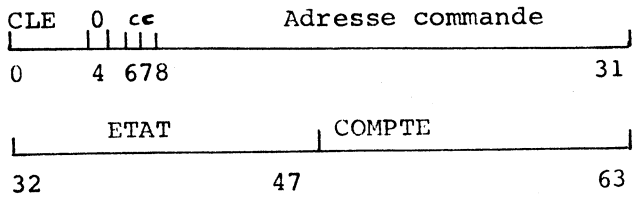
Opérations pour un programme canal contenant plusieurs commandes SEEK

Figure 3.10

Pour l'opération indiquée sur la figure 3.10, une seule instruction SIO est émise pour toute l'opération, et le multiplexage peut être fait pendant toutes les opérations de positionnement du bras.

3.22 Instruction SIOF (SIO Fast release)

L'instruction SIOF permet d'initaliser les opérations d'entrée-sortie de la même façon qu'une instruction SIO. Cependant, lorsqu'elle est exécutée pour un canal multiplex par bloc, et que le multiplexage est autorisé pour l'opération qui va être lancée, aucune vérification sur l'unité n'est faite pendant l'exécution de cette instruction. Toute anomalie est indiquée ultérieurement par une interruption différée. Le code condition dont l'indication est différée par SIOF est indiqué lors de cette interruption dans les bits 6 et 7 du mot d'état canal (figure 3.11).



Mot d'état canal du 370

Figure 3.11

Cette instruction permet de retarder le traitement de conditions particulières sur l'unité et de lancer immédiatement une nouvelle opération sur le canal libre.

3.23 Relance automatique

Certains canaux ont la possibilité d'effectuer automatiquement la relance d'un programme canal en cas d'erreur, sans qu'il n'y ait d'interruption d'entrée-sortie. Les procédures de recouvrement d'erreur du superviseur d'entrée-sortie peuvent alors être simplifiées. Par exemple, sur les disques IBM 3330, lorsqu'il y a une erreur de données, le canal réexécute automatiquement l'opération. Sur les disques IBM 2314, une telle erreur est indiquée par interruption et le superviseur d'entrée-sortie doit relancer lui-même l'opération.

3.3 Entrées-sorties pour les machines virtuelles

Nous avons vu dans le chapître 1 comment le système CP67 génère des unités d'entrée-sortie pour une machine virtuelle. Nous allons examiner quelques problèmes particuliers propres aux entrées-sorties réalisées dans des systèmes générant des machines virtuelles sur des ordinateurs disposant de la pagination.

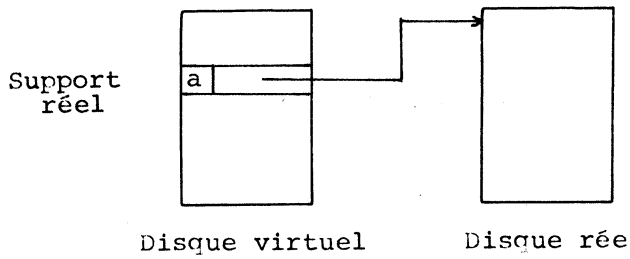
3.31 Traduction des programmes canaux

Une demande d'entrée-sortie virtuelle est une suite de commandes adressées à une unité virtuelle et relatives chacune à un bloc de données. Un bloc de données est déterminé par une adresse virtuelle en mémoire centrale et une longueur. Le bloc de données peut éventuellement être vide. Dans le système IBM/360, une demande d'entrée-sortie correspond à un programme canal.

Convertir une demande d'entrée-sortie virtuelle en une demande réelle revient à remplacer les commandes et les adresses de la commande virtuelle en commandes et adresses équivalentes pour la demande réelle. Si l'unité réelle et l'unité virtuelle sont de même type, la conversion des commandes virtuelles en commandes réelles ne pose pas de problème, puisque ces deux unités admettent les mêmes commandes.

Le transfert d'un bloc de données entre la mémoire centrale et une unité périphérique se fait d'un emplacement d'origine à un emplacement de destination. L'un de ces emplacements est connu par son adresse en mémoire centrale, l'autre est un enregistrement sur une unité d'entrée-sortie. Les adresses qui repèrent ces deux emplacements sont virtuelles dans la demande d'entrée-sortie virtuelle. Comme les canaux

n'ont pas accès aux dispositifs de pagination, l'adresse en mémoire virtuelle doit être convertie en adresse en mémoire réelle. L'adresse réelle du bloc de données en mémoire centrale est obtenue très rapidement par utilisation de la table des pages (et table des segments s'il y a lieu) ou par exécution d'une instruction qui donne l'adresse correspondant à une adresse virtuelle (instruction LRA du 360/67). Si les pages contenant les données ou les commandes ne se trouvent pas en mémoire centrale, il faut les y amener. Ces pages doivent être verrouillées jusqu'à ce que l'entrée-sortie soit achevée. L'adresse réelle du bloc de données sur l'unité peut être retrouvée grâce à une table décrivant quels sont les supports réels d'une unité virtuelle (figure 3.12).



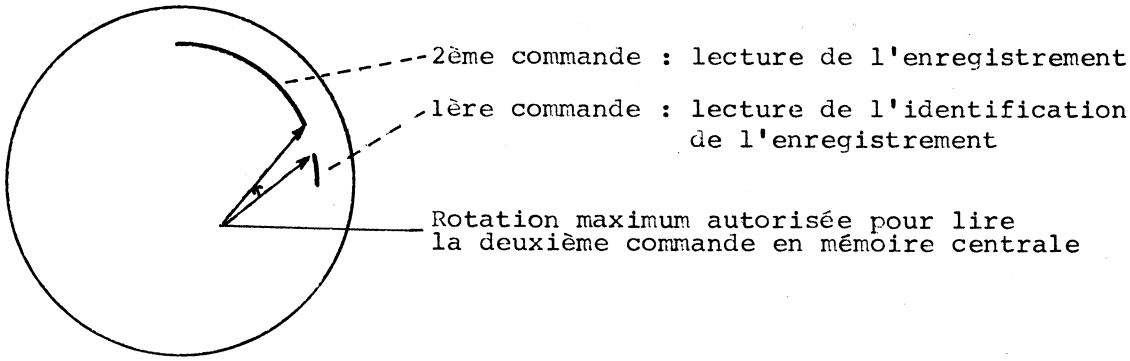
a : numéro du premier cylindre du disque virtuel sur son support réel

Liaison unité réelle-unité virtuelle

Figure 3.12

On peut se demander pourquoi les canaux n'ont pas accès à la pagination pour traduire automatiquement les adresses de données en mémoire centrale lorsqu'ils exécutent des commandes. Ce serait plus rapide. Mais il faut considérer qu'une entrée-sortie sur une unité mobile (disque, tambour) est un travail en temps réel, dès qu'il y a chaînage de commandes ou de données. L'exécution d'une commande peut avoir à se faire dans un temps limité après l'exécution de la commande précédente : par exemple, la première commande sert à localiser un

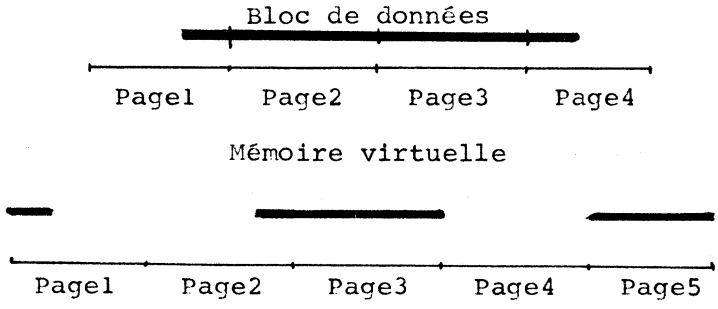
enregistrement, et la commande suivante à lire ou écrire cet enregistrement. Un défaut de page survenant pour la deuxième commande obligerait à exécuter ultérieurement une partie de la chaîne des commandes déjà effectuées, avec tous les risques de perte d'information que cela comporte.



Exemple d'opération d'entrée-sortie en temps réel

Figure 3.13

Nous avons considéré jusqu'à maintenant qu'à une commande virtuelle de transfert de données correspondait une commande réelle unique. Ce n'est pas toujours possible car un bloc de données peut chevaucher une ou plusieurs frontières de pages, et être éclaté en plusieurs sous-blocs non contigüs en mémoire réelle (figure 3.14).



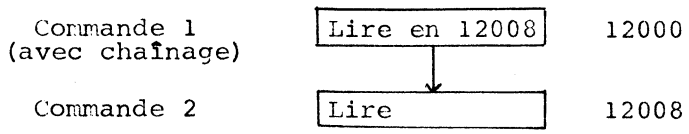
Bloc de données non continu en mémoire réelle

Figure 3.14

Comme le bloc de données doit être un enregistrement unique sur l'unité réelle, il faut substituer à la commande virtuelle une suite de commandes permettant de regrouper dans un seul enregistrement plusieurs sous-blocs dispersés en mémoire centrale. Sur les ordinateurs 360, le chaînage de données est utilisé dans ce but.
Certaines unités (p. e. tambours) sont trop rapides et ne laissent pas au canal le temps de lire une commande d'accès à un nouveau sous-bloc pendant l'accès à un enregistrement du tambour. Cet inconvenient est évité si le canal est conçu pour pouvoir lire de manière anticipée les adresses des sous-blocs de données. Ceci a été réalisé sur certains canaux des ordinateurs IBM/370 (dispositif d'adressage indirect des données par le canal).

Programmes canaux auto-modifiables

Un autre problème est créé lors de la conversion d'une demande d'entrée-sortie virtuelle : il est dû à l'emploi, dans certains systèmes, de programmes canaux auto-modifiables. Ces programmes canaux profitent du fait que le canal ne lit une nouvelle commande en mémoire centrale que lorsque la précédente a été exécutée. Cette dernière peut donc modifier la commande qui la suit (figure 3.15). Une telle modification peut par exemple permettre de préciser la longueur réelle d'un bloc de données à lire si cette longueur est enregistrée en tête du bloc. La première commande consiste à lire ce début de bloc pour garnir la partie longueur de la deuxième commande.
Lorsque c'est une demande d'entrée-sortie virtuelle, les nouvelles commandes lues sur unité sont des commandes virtuelles, et il faut les transformer en commandes réelles avant de les exécuter.



Programme canal auto-modifiable

Figure 3.15

Une telle demande est exécutée par chaînes de commandes qui ne soient pas auto-modifiables. Dès que la première chaîne non auto-modifiable a été exécutée, une nouvelle chaîne de longueur maxima et non auto-modififable est construite et exécutée. Ce cycle se poursuit jusqu'à ce que le programme canal virtuel soit achevé.

Un tel programme canal ne peut être exécuté sur une machine virtuelle que dans la mesure où il n'y a pas de contrainte de temps pour l'intervalle séparant l'exécution de deux demandes successives. De toute manière, cette auto-modification des programmes canaux devrait être à proscrire.

3.32 Utilisation de la mémoire et du temps

Un problème qui peut encore être soulevé à propos des entrées-sorties dans les machines virtuelles est celui de l'encombrement de la mémoire centrale. Les commandes et les données utilisées dans une demande d'entrée-sortie virtuelle sont verrouillées en mémoire centrale pendant toute la durée de vie de la demande. Si un programme canal unique travaille sur de grandes masses d'informations, il peut bloquer à son profit une part importante de la mémoire centrale. Le temps de verrouillage est d'autant plus long que le programme canal contient de commandes. D'autre part, si un programme canal boucle par suite d'une erreur de programmation, il peut monopoliser un canal et abaisser de façon importante le rendement du

système. Pour éviter ces inconvénients, on peut limiter le nombre de pages que peut verrouiller un utilisateur et implanter un "chien de garde" qui note les heures de lancement des demandes d'entrée-sortie et inspecte périodiquement les files d'attente sur les unités : les demandes actives depuis un temps excessif sont alors stoppées. Le "chien de garde" peut aussi être utilisé, indépendamment de la pagination, pour réinitialiser l'activité sur une unité ou un canal lorsqu'une interruption s'est perdue par suite d'une erreur du "hardware" (réf ESOES).

3.33 Evolution des entrées-sorties dans les machines virtuelles

La simulation complète des entrées-sorties par un système générateur de machines virtuelles présente dans certains cas de très grands avantages. Il en est ainsi par exemple lorsqu'un nouveau système doit être mis au point. Cependant, la plupart des utilisateurs n'ont pas besoin d'une machine virtuelle "pure". Il leur suffirait de pouvoir accéder à des enregistrements sans avoir à se préoccuper du déroulement physique de l'entrée-sortie. D'autre part, un système générateur de machines virtuelles crée une redondance dans le traitement des entrées-sorties. Un programme canal est d'abord construit par un programme de la machine virtuelle pour une unité virtuelle ; il est ensuite analysé et un programme canal destiné à l'unité réelle correspondante est construit. Réciproquement, les interruptions créées par les unités réelles doivent être analysées puis converties en interruptions pour les unités virtuelles. Comme le système générateur de machines virtuelles doit garder le contrôle des ressources d'entrée-sortie partagées, on crée des machines virtuelles "dégénérées" qui permettent d'accéder aux enregistrements par des ordres excluant toute notion d'asynchronisme et d'interruption (réf

BH). Ce type d'accès a été implanté dans GMS pour l'utilisation de l'imprimante, du lecteur et du perforateur de cartes (2.5).

Dans les systèmes conçus pour utiliser la pagination, on utilise des fichiers dont le format d'un enregistrement est une page. Le même mécanisme peut alors être utilisé pour accéder aux enregistrements des fichiers et pour accéder aux unités utilisées comme extension de la mémoire centrale. L'accès à un enregistrement de fichier se fait seulement par une modification de la table des pages et de la table qui repère les pages virtuelles sur la mémoire auxiliaire. L'enregistrement est amené en mémoire centrale par le programme qui traite les défauts de page seulement lorsqu'un élément de cet enregistrement est référencé. Les enregistrements non référencés ne sont pas transférés en mémoire centrale.

Cette solution a été employée dans plusieurs systèmes. Un tel mécanisme était inclus dans la conception du système ESOPE (1.5). Des modifications faites à la version de CP67 qui tourne à l'Université de Grenoble l'ont aussi inclus dans ce système (réf XL). Cette structuration des fichiers ne supprime cependant pas totalement les entrées-sorties puisque tout accès à une unité périphérique se fait finalement par une opération d'entrée-sortie.

Conclusion

La réalisation d'un superviseur d'entrée-sortie serait bien simplifiée si le fonctionnement des unités d'entrée-sortie était beaucoup plus normalisé qu'il ne l'est actuellement. Il faudrait, pour cela, définir quelques fonctions de base communes à toutes les unités d'un type donné ; ces fonctions pourraient être utilisées indistinctement sur toutes les unités de ce type.
Par exemple, pour les terminaux de type machine à écrire, deux fonctions permettraient de LIRE ou d'ECRIRE une ligne. De plus, tous ces terminaux pourraient avoir un dispositif normalisé permettant de créer une interruption asynchrone indépendamment des opérations lancées sur l'unité. Actuellement, c'est le superviseur d'entrée-sortie qui joue ce rôle de normalisation. Il faudrait donc ramener au niveau du "hardware" (à l'intérieur des unités ou des unités de contrôle) une partie du travail effectuée actuellement par le superviseur d'entrée-sortie.

CONCLUSION

L'objectif du travail que nous venons de présenter était de concevoir et de réaliser un système complet d'entrée-sortie dans le cadre du projet GMS. Cet objectif a été atteint et le système GMS est maintenant opérationnel. Cette réalisation nous a permis de mesurer toutes les difficultés qui attendent le programmeur qui veut gérer lui-même des unités d'entrée-sortie. Ces difficultés sont mises en évidence dans les systèmes qui simulent des entrées-sorties ; nous en avons vu un exemple avec CP67. Il est certain que la normalisation et la simplification du fonctionnement des unités d'entrée-sortie résoudraient en grande partie ces difficultés. Les fonctions que pourrait remplir une unité feraient partie d'un jeu restreint de commandes universelles. Il serait aussi très intéressant que les unités puissent s'occuper elles-mêmes des protections d'accès. Puisque plusieurs systèmes réalisent les entrées-sorties sur certaines unités par transfert de pages entières, on peut envisager de disposer sur les unités des clés de protection associées à chaque enregistement et utilisables de la même manière que les clés de protection en mémoire centrale.

Quoiqu'il en soit, les entrées-sorties continueront de jouer un rôle primordial dans les systèmes. Un superviseur d'entrée-sortie offrant suffisamment de possibilités aux programmes qui l'utilisent pourrait être employé dans plusieurs systèmes différents sur une même machine. Inversement, si on implante sur plusieurs ordinateurs differents un superviseur d'entrée-sortie identique à l'exception des séquences qui contrôlent directement le hardware, il devrait être possible de rendre un système indépendant de l'ordinateur qui le supporte vis-à-vis des entrées-sorties : c'est un objectif dont la réalisation pourrait mener vers l'indépendance des programmes vis-à-vis des machines.

ANNEXE

FONCTIONNEMENT DES ENTREES-SORTIES

SUR IBM/360

A.1 Fonctionnement de l'unité centrale

A.11 Mot d'état programme PSW

L'état de l'unité centrale détermine, à un instant donné, la prochaine instruction à exécuter et les conditions dans lesquelles elle doit être exécutée. Il est représenté en permanence par un double mot, le mot d'état programme (PSW : Program Status Word) Il ne peut être lu par un programme qu'à la suite d'une interruption. Mais il peut être modifié par l'instruction privilégiée LPSW qui remplace l'ancien PSW par un nouveau. Le format du mot d'etat programme est indiqué à la figure 1.

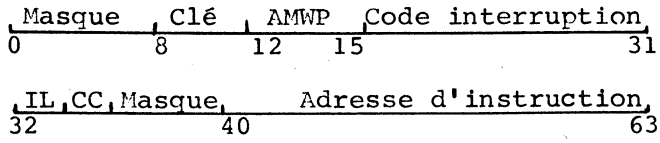


Figure 1 Mot d'état programme

Bits 0-7 Masque : à chacun des bits 0-6 correspond un canal (A.2)
Au bit 7 correspond l'interruption externe
Valeur du bit 0 : les interruptions en provenance de cette source sont suspendues.
Valeur du bit 1 : les interruptions en provenance de cette source sont autorisées.

Bits 8-11 CLE : la mémoire centrale étant divisée en blocs de 2048 octets, protegés chacun par une clé de 4 bits, ces blocs ne peuvent être modifiés que si la clé du mot d'état programme est identique à la clé de protection du bloc.

Bit 12 Ce bit indique le code employé par l'unité centrale
Bit 13 Ce bit est un masque pour les interruptions "Erreur machine"
Bit 14 Attente : Ce bit vaut 1 quand l'unité centrale est inactive
Bit 15 Problème : indique le mode de travail de l'unité centrale
(0 : mode maître, 1 : mode esclave).

Bits 16-31 Code interruption : ces bits contiennent de l'information significative après interruption.

Bits 32-33 Ces 2 bits indiquent la longueur en demi-mots de la dernière instruction exécutée (un demi-mot=2 octets)

Bits 34-35 Code condition : ces 2 bits sont positionnés après certaines instructions et leur valeur est utilisée dans les instructions de branchement. Par exemple, ils sont positionnés dans les opérations arithmétiques pour indiquer si le résultat est positif, négatif ou nul.

Bits 36-39 Ces bits permettent de suspendre les conditions d'interruptions survenant après certaines opérations de calcul.

Bits 40-63 Ces trois octets indiquent l'adresse de la prochaine instruction à exécuter.

A.12 Interruptions

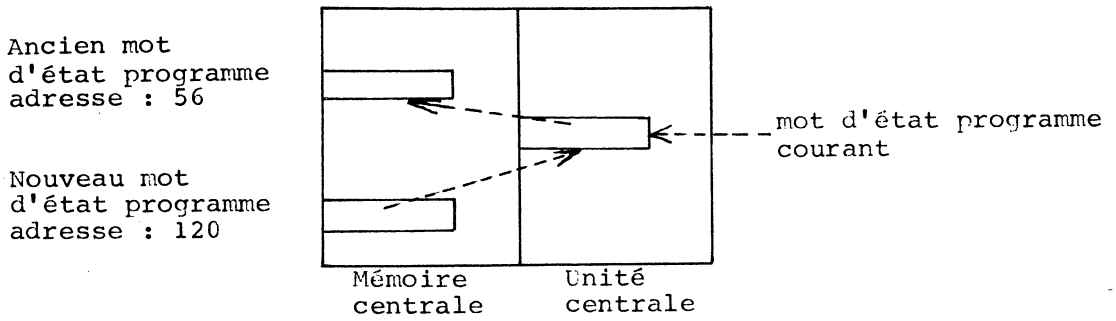
Une interruption est un évènement qui sauvegarde l'état courant de l'unité centrale et le remplace par un nouvel état. Il y a cinq classes d'interruptions ; pour chaque classe, il existe en mémoire centrale, à des emplacements qui sont les mêmes sur tous les ordinateurs 360, un double mot réservé pour la sauvegarde du mot d'état programme au moment de l'interruption, et un double mot contenant le nouveau mot d'état programme. Les cinq classes sont les suivantes :

- interruption externe
- interruption SVC
- interruption programme
- interruption erreur machine
- interruption d'entrée-sortie

Le système peut suspendre l'apparition des interruptions externes, des interruptions d'erreur machine et des interruptions d'entrée-sortie en mettant le bit correspondant à 1 dans le mot d'état programme : les interruptions sont alors conservées dans les circuits cablés : elles sont pendantes. elles surviennent dès qu'une modification du mot d'état programme met le bit correspondant à 1.

Etant donné qu'il existe un seul emplacement pour sauvegarder le mot d'etat programme par classe d'interruption, une interruption détruit le mot d'état programme sauvé par l'interruption précédente de la même classe. Aussi le programmeur doit transférer dans des zones qui lui soient propres ces mots d'état programme, si une nouvelle interruption peut survenir. Il peut aussi se protéger en suspendant les interruptions d'une classe quand c'est possible.

D'autre part, les registres n'étant pas sauvegardés automatiquement par une opération cablée, ils doivent l'être par le système explicitement, avant qu'ils ne soient modifiés.



Phase 1 : Le mot d'état programme de l'unité centrale est rangé à l'adresse 56

Phase 2 : Le mot d'état programme pour interruption d'entrée-sortie situé à l'adresse 120 est chargé dans le mot d'état programme de l'unité centrale.

Exemple : commutation cablée des mots d'état programme créée par une interruption d'entrée-sortie

Figure 2

A.2 Fonctionnement des unités d'entrée-sortie

A.21 Canaux-Unités de contrôle-Unités

A.211 Définitions

Une opération d'entrée-sortie implique le transfert d'information entre la mémoire centrale et une unité d'entrée-sortie : lecteur de cartes, terminal, disque, ... ; elle est dirigée par l'unité de contrôle à laquelle est rattachée l'unité (figure 3).

L'unité de contrôle fournit les fonctions logiques nécessaires pour faire travailler et contrôler une ou plusieurs unités d'entrée-sortie et elle adapte les caractéristiques de chaque unité à la forme standard des informations envoyées ou reçues par le canal.

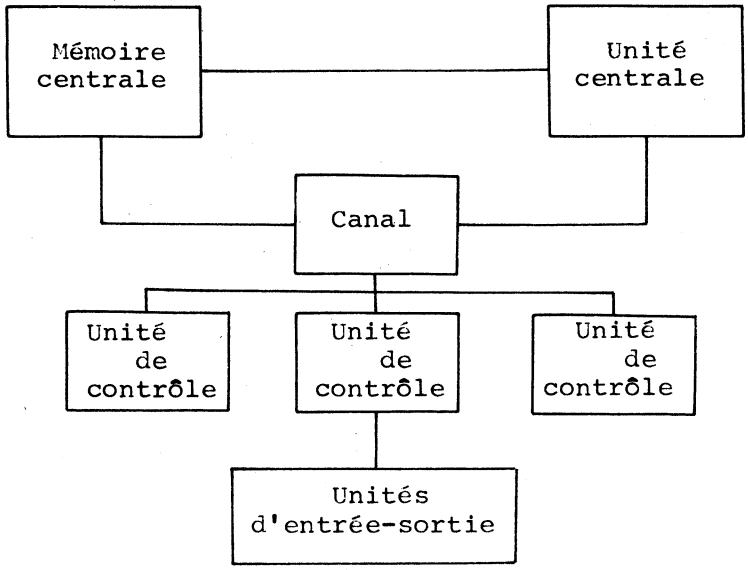


Schéma des connexions
dans un ordinateur 360

Figure 3

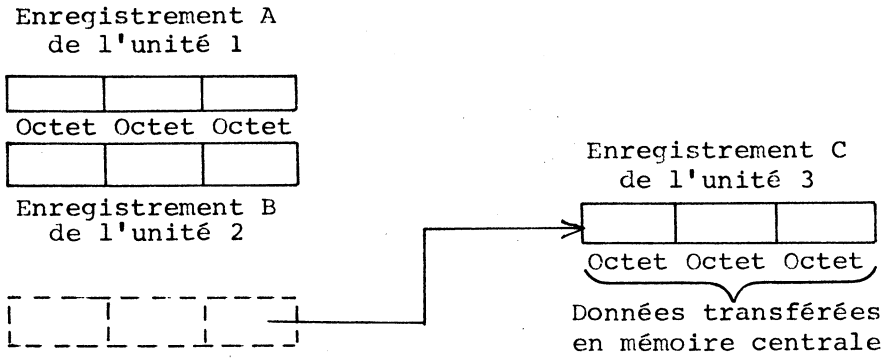
Le canal décharge l'unité centrale de la communication directe avec les unités d'entrée-sortie, et il permet à l'unité centrale de fonctionner en simultanéité avec les unités d'entrée-sortie.

Il exécute des instructions qui sont des "commandes canal". Physiquement, le canal peut partager le "hardware" avec l'unité centrale, ou disposer d'un équipement entièrement autonome. Ceci influe uniquement sur l'efficacité du système et ne modifie pas la programmation.

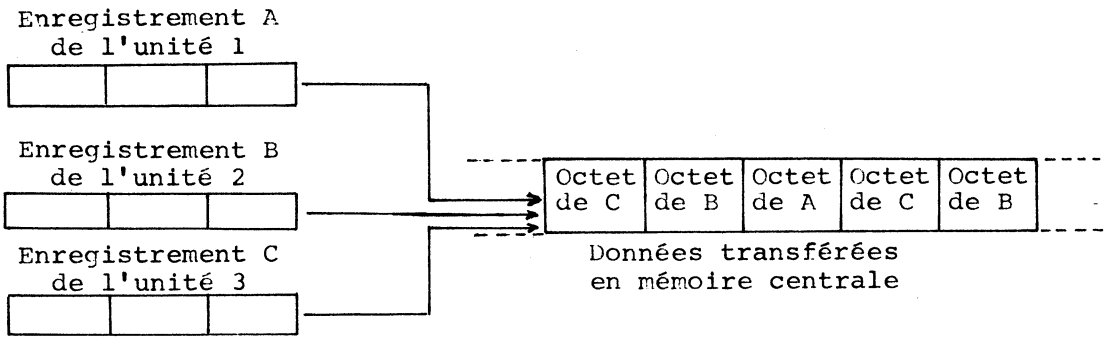
A.212 Modes de transfert

Le transfert des données entre la mémoire principale et une unité d'entrée-sortie peut se faire en mode continu ou en mode discontinu. En mode continu, l'unité d'entrée-sortie monopolise le canal pendant tout le transfert d'un groupe d'informations. Aucune autre unité ne peut communiquer avec le canal pendant ce transfert. En mode discontinu, les ressources du canal sont partagées par plusieurs unités d'entrée-sortie travaillant simultanément. Dans ce mode, toutes les unités d'entrée-sortie utilisent le canal pendant de courtes périodes de temps durant lesquelles seul un segment d'information est transféré. Un segment d'information peut consister en un seul octet, quelques octets, une indication d'état de l'unité, ou une séquence de contrôle utilisée pour l'initiation d'une nouvelle opération.

Un canal qui fait travailler une unité en mode continu ne peut accepter de nouvelles opérations d'entrée-sortie, alors qu'un canal travaillant en mode discontinu sur une unité d'entrée-sortie peut en démarrer sur d'autres unités.



Mode continu



Mode discontinu

Modes de transfert des canaux

Figure 4

La distinction entre ces deux modes est très importante. Les modules qui gèrent les entrées-sorties doivent en tenir compte pour ne pas bloquer le système en essayant de partager un canal qui ne peut opérer qu'en mode continu.

A.213 Types de canaux

Il existe deux types de canaux qui se différencient par le mode de travail qu'ils peuvent supporter. Un canal multiplex peut opérer en mode continu ou discontinu suivant l'unité desservie. Un canal sélecteur ne peut opérer qu'en mode continu.

On appelle sous-canal les ressources mémoires du canal nécessaires au déroulement d'une opération d'entrée-sortie. Cette mémoire qui est propre au canal contient l'adresse et la longueur des données transférées, ainsi que l'information d'état et de contrôle associée a l'opération d'entrée-sortie.

Le mode dans lequel un canal peut travailler dépend du nombre de ses sous-canaux.

Un canal sélecteur a un seul sous-canal et force toujours l'unité d'entrée-sortie à travailler en mode continu. Il ne peut être occupé qu'à un transfert de données à la fois. Mais, dans le même temps, d'autres unités d'entrée-sortie attachées au canal peuvent exécuter des opérations initialisées antérieurement et n'impliquant plus de communication avec le canal. C'est le cas par exemple du rebobinage d'une bande ou du positionnement du bras d'accès d'un disque.

Le canal multiplex contient plusieurs sous-canaux et peut opérer en mode continu ou discontinu. En mode multiplex, une ou plusieurs unités peuvent transférer des données concurrement, chacune sur un sous-canal distinct. En mode continu, une seule opération d'entrée-sortie peut se dérouler pour le sous-canal.

Le mode d'opération est déterminé par l'unité d'entrée-sortie. En général, les unités ayant un grand débit opèrent avec le canal en mode continu (disques, bandes), et les autres unités opèrent avec le canal en mode discontinu.

A.214 Adresse d'unité

Au niveau de l'ordinateur, une unité d'entrée-sortie est toujours connue par son adresse. Celle-ci est composée de trois symboles héxadécimaux. Le premier indique le numéro du canal et les deux autres indiquent les numéros de l'unité de contrôle et de l'unité tels qu'ils sont connus par le canal.

Exemple : L'unité d'adresse 041 est connectée au canal 0, et elle est reconnue comme ayant l'adresse 41 par l'unité de contrôle à laquelle elle est rattachée.

A.22 Déroulement d'une opération d'entrée-sortie

A.221 Schéma global d'une opération d'entrée-sortie

Une opération d'entrée-sortie est l'exécution par le canal d'un programme canal, séquence de commandes canal. Elle est initialisée par une instruction privilégiée SIO. L'opération peut être rejetée à l'exécution de cette instruction. Un traitement approprié s'impose alors. Si l'opération est acceptée, l'activité de l'unité centrale reprend en même temps que se déroule l'entrée-sortie. Le programmeur peut mettre l'unité centrale en attente de la fin de l'opération. La fin de celle-ci est indiquée par une ou plusieurs interruptions d'entrée-sortie. Le mot d'état programme est rangé à l'adresse 56, ainsi que le mot d'état canal à l'adresse 64. Le mot d'état programme contient dans la partie "code interruption" l'adresse physique de l'unité qui interrompt. Le nouveau mot d'état programme est pris à l'adresse 120. Ce dernier, initialisé par le programmeur, donne l'adresse d'un module de traitement des interruptions d'entrée-sortie.

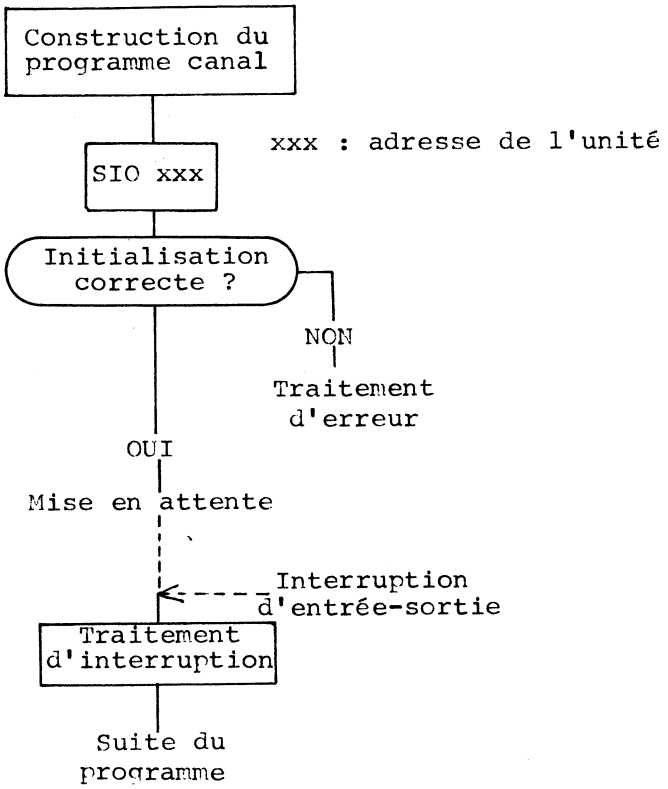
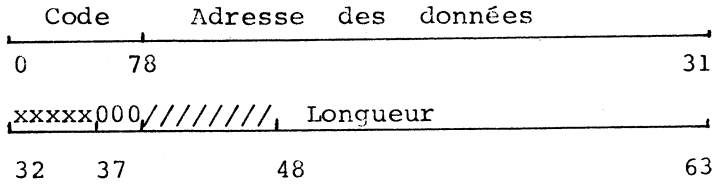


Schéma d'une opération d'entrée-sortie

Figure 5

A.222 Commandes canal - Mot d'adresse canal

Le programme canal est une séquence de commandes canal (CCW : Channel Command Word) qui se trouvent en mémoire principale. A l'initialisation de l'opération d'entrée-sortie , l'adresse de la première commande doit se trouver dans le mot d'adresse canal (CAW : Channel Address Word) (figure 5). Il est à un emplacement fixe en mémoire centrale, à l'adresse 72. Les bits 0-3 de ce mot contiennent une clé de protection : pendant toute la durée de l'opération, les seuls blocs de 2048 octets en mémoire centrale dans lesquels le canal puisse transférer des données sont ceux dont la clé de protection est identique à celle contenue dans le mot d'adresse canal.



Commande canal

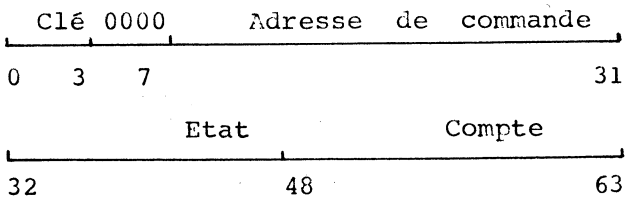
Figure 7

Une commande canal occupe un double mot en mémoire centrale (figure 7). Elle est lue par le canal qui l'interprète et la transmet à l'unité de contrôle. Elle se rapporte toujours à un enregistrement physique sur l'unité ou une partie d'enregistrement physique, lorsqu'il y a transfert de données. Il n'est pas possible par exemple de lire deux cartes perforées, avec une seule commande canal.
Le code de la commande est indiqué dans le premier octet de la commande. Il existe six types de commande : lecture, lecture arrière, écriture, contrôle, anayse, branchement. Les commandes de lecture, lecture arrière et ecriture transfèrent des enregistrements entre la mémoire centrale et l'unité. Les commandes de contrôle indiquent des opérations spécifiques à une unité : rebobiner une bande, avancer le papier sur l'imprimante, ... La commande d'analyse fournit des informations détaillées sur l'unité d'entrée-sortie en cas d'erreur. La commande de branchement provoque une rupture de séquence dans le programme canal.
L'adresse du bloc de données à transférer se trouve dans les bits 8-31, et la longueur de ce bloc dans les bits 48-63.
Divers indicateurs occupent les bits 32-36 :
Le bit 32 indique le chaînage de données quand il a la valeur 1 : quand la commande courante est achevée, le canal reexécute la même commande, en prenant dans le double mot suivant l'adresse et la longueur des données à transférer.
Le bit 33 indique le chaînage de commandes quand il a la valeur 1 : quand la commande courante est achevée, le canal exécute la commande qui se trouve dans le double mot suivant.
Le bit 34, quand il a la valeur 1, supprime l'interruption qui se produit quand la longueur indiquée dans le mot de commande est différent de celle de l'enregistrement physique lu.
Le bit 35, significatif seulement pour les commandes de lecture et d'analyse, entraîne la suppression du transfert réel des informations

en mémoire centrale.
Quand le bit 36 a la valeur 1, une interruption est créée à la fin de l'exécution de la commande, même s'il y a chaînage de commandes ou de données.

A.223 Mot d'état canal

Le mot d'état canal fournit au programme l'état d'une unité d'entrée-sortie ou les conditions dans lesquelles une opération d'entrée-sortie s'est terminée. Il est rangé en mémoire centrale à l'adresse 64, à chaque interruption d'entrée-sortie, et il peut aussi l'être à l'exécution des instructions SIO, HIO et TIO. Dans le cas d'une interruption marquant la libération d'un sous-canal, il permet de retrouver la dernière commande exécutée du porogramme canal. La figure 8 indique le format du mot d'état canal.



Mot d'état canal

Figure 8

Bits 0-3 : clé de protection utilisée pour l'opération d'entrée-sortie
Bits 4-7 : nuls
Bits 8-31 : pour une interruption marquant la libération du sous-canal, cette zone contient l'adresse de la dernière commande exécutée, augmentée de 8. Dans les autres cas, elle contient une valeur nulle lors d'une interruption, et une valeur quelconque si le mot d'état canal est rangé à l'exécution d'une instruction d'entrée-sortie.
Bits 32-47 : ces bits identifient les conditions sur l'unité et sur le canal qui ont causé le rangement du mot d'état canal.

Bit 32 : Attention : cette indication est mise pour certaines unités, quand elles détectent une condition d'interruption indépendante de toute opération d'entrée-sortie. (par exemple l'opérateur appuie sur la touche "Attention" du pupître).
Bit 33 Modificateur d'état : il indique que l'unité de contrôle est occupé ou que la séquence normale des commandes doit être modifiée.
Bit 34 Fin sur unité de contrôle : il marque la libération de l'unité de contrôle.
Bit 35 Occupé : il indique que l'unité de contrôle ou l'unité ne peuvent exécuter la commande ou l'instruction car elles exécutent une opération initialisée antérieurement ou qu'elles contiennent une interruption pendante.
Bit 36 Fin sur canal : Cette indication apparaît pour toute fin normale d'entrée-sortie, et marque la libération du sous-canal. En cas de fin anormale, il arrive que cette indication soit supprimée.
Bit 37 Fin sur unité : cette indication est mise quand une opération d'entrée-sortie est terminée sur l'unité ou quand certaines unités sont mises dans l'état "prêt".

Bit 38 Erreur sur unité : l'unité a eu une anomalie de fonctionnement. Il faut alors lancer une commande d'analyse pour connaître plus précisement cette anaomalie. Une telle interruption se produit par exemple lorsque l'unité est mise dans l'état "non prêt" au cours d'une opération d'entrée-sortie.
Bit 39 Exception sur unité : il indique une situation anormale, mais il ne reflète pas toujours une erreur à corriger : par exemple, le magasin d'un lecteur de cartes est épuisé.

Les bits 40-47 indiquent des interruptions dont les conditions sont détectées par le canal.

A.224 Instructions d'entrée-sortie

Quatre instructions privilégiées permettent au programmeur d'agir sur les unités d'entrée-sortie.

Une opération d'entrée-sortie est initialisée par l'instruction SIO. Pendant l'exécution de cette instruction, l'unité centrale s'assure que le canal desservant l'unité est libre. Le canal lit alors le mot de commande canal dont l'adresse est dans le mot de commande canal, vérifie sa validité et initialise l'opération sur l'unité, si elle est libre. Le code condition du mot d'état programme est positionné, et l'activité de l'unité centrale reprend à l'instruction suivant SIO. Le code condition peut prendre les valeurs 0, 1, 2, 3.
Le code condition 0 indique que l'opération s'est déroulée correctement.
Le code condition 1 indique qu'un mot d'état canal CSW a été rangé :
-L'opération spécifiée dans le mot de commande canal est une opération immédiate, et il n'y a pas de chaînage. Une opération immédiate n'implique pas de transfert d'information entre la mémoire centrale et le canal autre que le mot de commande canal (par exemple rebobiner une bande). Le canal est libéré avant la fin de l'instruction SIO, tandis que l'unité exécute la fonction demandée.
-L'unité ou l'unité de contrôle à laquelle elle est rattachée est occupée ou contient une interruption pendante : l'interruption est supprimée, et le mot d'état canal identifie l'interruption. L'opération d'entrée-sortie n'est pas initialisée.
-L'unité ou l'unité de contrôle est occupée, ou l'unité de contrôle contient une interruption pendante pour une autre unité que celle adressée par l'instruction. L'opération d'entrée-sortie n'est pas initialisée.
-L'unité ou le canal a detecté une erreur de fonctionnement ou de programmation. Le mot d'état canal identifie la condition d'erreur.
Le code condition 2 indique que le canal ou le sous-canal est occupé, ou qu'ils contiennent une interruption pendante.
Le code condition 3 indique que l'adresse d'unité n'est pas celle d'une unité rattachée à la configuration, ou que l'unité de contrôle est hors fonction.

L'instruction TIO permet de tester l'état d'une unité ou d'un canal en positionnant le code condition du mot d'état programme, et dans certaines conditions en rangeant le mot d'état canal. Les interruptions pendantes peuvent être supprimées.

L'instruction TCH permet de tester l'état d'un canal : disponible, contenant une interruption pendante, ou occupé.

L'instruction HIO permet d'arrêter une opération initialisée antérieurement sur cette unité.

A.225 Exemple d'opération d'entrée-sortie

L'opération d'entrée-sortie qui suit se déroule sur une imprimante. C'est l'enchaînement de deux commandes canal : la première fait avancer le papier d'une ligne, la deuxième écrit une ligne et avance le papier d'une ligne. Il faut d'abord que le nouveau mot d'état programme d'entrée-sortie soit initialisé. Cette opération est généralement faite une seule fois à l'initialisation du système.

```
        MVC   120(8), NOUVPSW initialisation du nouveau PSW
        MVC   72(4), APROGCAN initialiser mot d'adresse canal
        LH    8, APRINT  chargement de l'adresse de l'imprimante
        SSM   =X'00' empêcher les interruptions
                     sur tous les canaux
SIO     SIO   0(8) initialisation de l'opération
        BC    2, SIO canal occupé : reessayer SIO
        BC    1, NOTOPER Unité non opérationnelle
        BC    4, CSWSTORD CSW rangé
        LPSW  ATTENTE Initialisation correcte,
                      mettre l'unité centrale en attente


NOTOPER I
        I                Séquence prévenant l'opérateur
        I                que l'unité n'est pas opérationnelle
        I
CSWSTORD I
        I                Séquence analysant le CSW rangé
        I
        I
SUITE   I
        I          Suite du programme
        I
```

Traitement des interruptions de l'imprimante

```
CSW     EQU   64 adresse du mot d'état canal
INTIMPR CLC   56+2(2), APRINT adresse de l'unité
                        dans l'ancien PSW d'entrée-sortie
        BNE   OUT  Branchement si ce n'est pas l'imprimante
        TM    CSW+4, B'11010011' Y a-t-il autre chose
                                 que fin sur canal,
                                 fin sur unité
                                 ou fin sur unité de contrôle ?
        BNZ   ERREUR Si oui, traiter les erreurs
        TM    CSW+4, X'04' Y a-t-il fin sur unité?
        BO    OK  Si oui, opération terminée
        LPSW  ATTENTE Si non, attendre fin sur unité
```

```
OK       B      SUITE  Opération terminée, passer à la suite
ERREUR   I
         I          Séquence de traitement d'erreur
         I
OUT      I
         I          Traitement des interruptions d'autres unités
         I
```

Nouveau PSW d'entrée-sortie, interruptions d'entrée-sortie interdites

```
         DS     0D
NOUVPSW  DC     X'01', X'04', X'0000', AL3(INTIMPR)
```

PSW d'attente, toutes interruptions autorisées

```
ATTENTE  DC     X'FF', X'06', X'0000', F'0'
         DS     0H
APRINT   DC     X'000E' Adresse physique de l'imprimante
APROGCAN DC     A(PROGCAN) Adresse du programme canal
```

Commande faisant avancer le papier d'une ligne

```
PROGCAN  CCW    X'0B', 0, X'60', 1
```

Commande faisant imprimer une ligne de données

```
         CCW    X'09', LIGNE, x'20', 30
LIGNE    DC     CL30'CETTE LIGNE DOIT ETRE IMPRIMEE'
```

BIBLIOGRAPHIE

AH A. AUROUX, C. HANS
Introduction aux systèmes CP67 et CMS ;
Les systèmes conversationnels, L. Bolliet
Monographies d'Informatique
Dunod

BH J. BELLINO, C. HANS
Virtual machine or virtual operating system ?
Workshop on Virtual Computers, Harvard 1973

BP J. BELLINO, Ph. POTIN
Mécanismes d'un hyperviseur ;
Congrès AFCET, Grenoble 1972

CK C. KAISER
Conception et réalisation de systèmes à accès multiple : gestion du parallélisme ;
Thèse d'Etat, Paris 1973

CMS IBM Corp.
CMS Program Logic Manual ;
Form GY20-0591

CP1 IBM Corp.
CP67/CMS User's Guide ;
Form GH20-0859

CP2 IBM Corp.
CP67 : Program Logic Manual
Form GY20-0590

EO E. ORGANICK
The MULTICS system : an examination of its structure ;
MIT Press

ESO C. BETOURNE, J. BOULENGER, J. FERRIE, C. KAISER, S. KRAKOWIAK, J. MOSSIERE
Le système ESOPE ;
Congrès AFCET, Paris 1970

ESOES G. BAUDET, J. FERRIE, C. KAISER, J. MOSSIERE
Entrées-sorties dans un système à mémoire virtuelle ;
Congrès AFCET, Grenoble 1972

EWD E. W. DIJKSTRA
The structure of the "THE" multiprogramming system ;
CACM Vol 11 No 5 May 1968

EY E. YOURDON
Design of on-line computer systems
Prentice Hall, 1972

FO R. FEIERTAG, E. ORGANICK
The MULTICS Input/Output System ;
Third Symposium on Operating System Principles,
Palo Alto, California, October 1971

IBM360 IBM Corp.
System 360 Principles of Operation ;
Form A22-6821

IBM370 IBM Corp.
System 370 Principles of Operation ;
Form GA22-7000

IBM67 IBM Corp.
Functional characteristics 360/67 ;
Form GA27-2719

JB J. BELLINO
Superviseur d'entrée-sortie dans un contexte de multiprogrammation ;
Cours du certificat "Conception de systèmes informatiques", Maîtrise d'informatique, Grenoble, Mai 1970

LF L. FINET
Gestion des ressources physiques dans un système avec contraintes de "swapping" ;
Séminaires de l'Institut de Mathématiques Appliquées, Grenoble, Janvier 1973

LH J.P. LE HEIGET
Généralisation de la notion d'espaces virtuels sous les systèmes CP67/CMS ;
Thèse CNAM, Grenoble, Juillet 1972

OMD J.F. OSSANA, L.E. MIKUS, S.D. DUNTEN
Communications and Input/Output Switching in a multiplex computing system ;
AFIPS, 1965 FJCC, Vol. 27, Part 1

OSIOS IBM Corp.
OS/360 Input/Output Supervisor Program Logic Manual ;
Form GY28-6616

OSSPG IBM Corp.
System Programmer's Guide ;
Form C28-6550

PJD P.J. DENNING
Virtual Memory ;
Computing Surveys, Vol. 2 No 3 Sept. 1970

SK S. KRAKOWIAK
Conception et réalisation de systèmes à accès multiple : allocation des ressources ;
Thèse d'Etat, Paris, 1973

XL X. DE LAMBERTERIE
Espaces virtuels et gestion de fichiers ;
Thèse de 3o cycle, Grenoble, 1973

BIBLIOGRAPHIE

classée par ordre alphabétique
des noms d'auteurs

A. AUROUX, C. HANS
Introduction aux systèmes CP67 et CMS ;
Monographies d'Informatique, L. Bolliet
Dunod

G. BAUDET, J. FERRIE, C. KAISER, J. MOSSIERE
Entrées-sorties dans un système à mémoire virtuelle ;
Congrès AFCET, Grenoble 1972

J. BELLINO
Superviseur d'entrée-sortie dans un contexte de multiprogrammation ;
Cours du certificat "Conception de systèmes informatiques", Maîtrise d'informatique, Grenoble, Mai 1970

J. BELLINO, C. HANS
Virtual machine or virtual operating system ?
Workshop on Virtual Computers, Harvard 1973

J. BELLINO, Ph. POTIN
Mécanismes d'un hyperviseur ;
Congrès AFCET, Grenoble 1972

C. BETOURNE, J. BOULENGER, J. FERRIE, C. KAISER,
S. KRAKOWIAK, J. MOSSIERE
Le système ESOPE ;
Congrès AFCET, Paris 1970

X. DE LAMBERTERIE
Espaces virtuels et gestion de fichiers ;
Thèse de 3o cycle, Grenoble, 1973

P.J. DENNING
Virtual Memory ;
Computing Surveys, Vol. 2 No 3 Sept. 1970

E. W. DIJKSTRA
The structure of the "THE" multiprogramming system ;
CACM Vol 11 No 5 May 1968

R. FEIERTAG, E. ORGANICK
The MULTICS Input/Output System ;
Third Symposium on Operating System Principles,
Palo Alto, California, October 1971

L. FINET
Gestion des ressources physiques dans un système avec contraintes de "swapping" ;
Séminaires de l'Institut de Mathématiques Appliquées, Grenoble, Janvier 1973

IBM Corp.
CMS Program Logic Manual ;
Form GY20-0591

IBM Corp.
CP67/CMS User's Guide ;
Form GH20-0859

IBM Corp.
CP67 : Program Logic Manual
Form GY20-0590

IBM Corp.
System 360 Principles of Operation ;
Form A22-6821

IBM Corp.
System 370 Principles of Operation ;
Form GA22-7000

IBM Corp.
Functional characteristics 360/67 ;
Form GA27-2719

IBM Corp.
OS/360 Input/Output Supervisor Program Logic Manual ;
Form GY28-6616

IBM Corp.
System Programmer's Guide ;
Form C28-6550

C. KAISER
Conception et réalisation de systèmes à accès multiple : gestion du parallélisme ;
Thèse d'Etat, Paris 1973

S. KRAKOWIAK
Conception et réalisation de systèmes à accès multiple : allocation des ressources ;
Thèse d'Etat, Paris, 1973

J.P. LE HEIGET
Généralisation de la notion d'espaces virtuels sous les systèmes CP67/CMS ;
Thèse CNAM, Grenoble, Juillet 1972

E. ORGANICK
The MULTICS system : an examination of its structure ;
MIT Press

J.F. OSSANA, L.E. MIKUS, S.D. DUNTEN
Communications and Input/Output Switching in a multiplex computing system ;
AFIPS, 1965 FJCC, Vol. 27, Part 1

E. YOURDON
Design of on-line computer systems
Prentice Hall, 1972