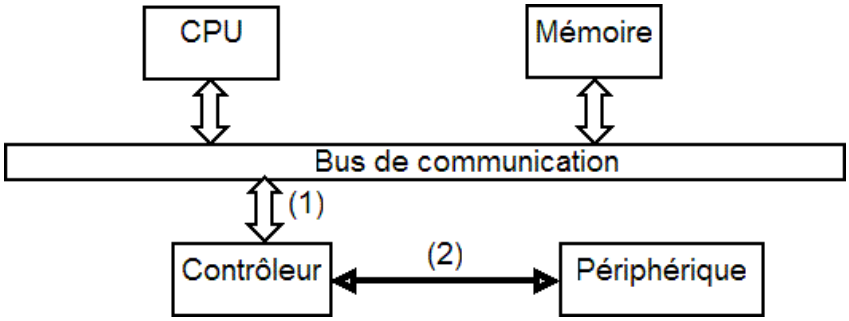
Systèmes d'Exploitation

Gestion des entrées / sorties

Version 1



Y. Challal, H. Bettahar, M. Vayssade

Table des matières

Objectifs

- Analyser le sous-système de gestion des entrées / sorties

Gestion des Entrées / Sorties

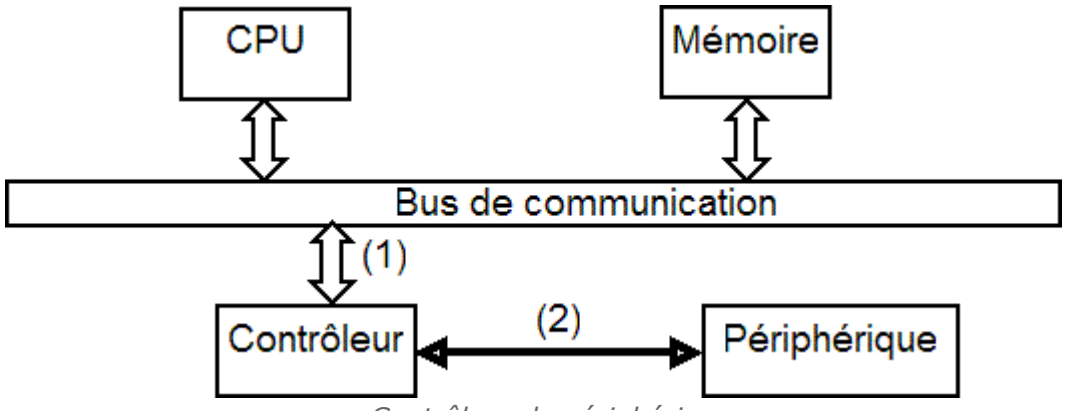
I

L'une des principales fonctions d'un système d'exploitation consiste à contrôler tous les périphériques d'entrée/sortie (E/S) de l'ordinateur. Il doit émettre des commandes vers les périphériques, intercepter les interruptions et gérer les erreurs. Il fournit également une interface simple entre les périphériques et le reste du système. Dans la mesure du possible, l'interface doit être la même pour tous les périphériques (indépendance des périphériques) *[tanenbaum03]*.

A. Aspects physiques des Entrées / Sorties

Définition : Contrôleur de périphérique

Chaque périphérique est connecté à l'ordinateur par l'intermédiaire d'une carte électronique spéciale appelée **contrôleur de périphérique**. La figure (cf. 'Contrôleur de périphérique' p 7) illustre une architecture "classique" de matériel.



Contrôleur de périphérique

Le contrôleur est chargée de transformer les signaux électriques (2) "compris" par le périphérique en signaux (1) compris par l'unité centrale ou la mémoire (et vice versa). Un contrôleur possède en général :

- une zone tampon (buffer),

- des bits de contrôle : bit d'état (prête, occupée), et bit de commande (lecture, écriture).

Communication contrôleur / processeur

Chaque contrôleur possède quelques registres qui servent à la communication avec le processeur. En écrivant dans ces registres, le système d'exploitation ordonne au périphérique de délivrer des données, d'en accepter, de s'activer ou se désactiver lui même ou encore d'effectuer une opération donnée. En lisant ces registres, le système d'exploitation peut connaître l'état du périphérique, savoir s'il est prêt à accepter une nouvelle commande, etc. Il existe deux possibilités de communication entre le processeur et les registres de contrôle :

Instructions spéciales : chaque registre de contrôle se voit assigner un numéro de porte d'E/S. Avec une instruction spéciale comme :

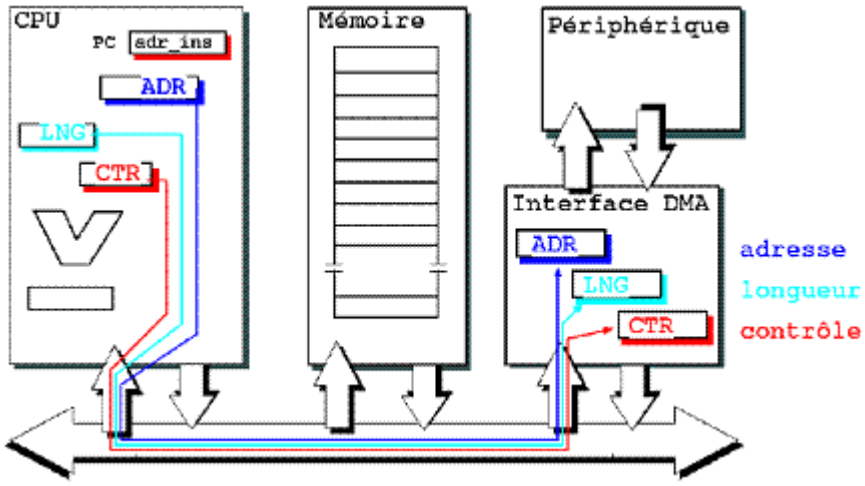
- **IN REG,PORT** : le processeur lit dans le registre PORT du contrôleur et stocke le résultat dans le REG du registre du processeur. De même, avec
- **OUT PORT,REG** : le processeur écrit le contenu de REG dans un registre PORT du contrôleur.

Cartographie mémoire : la seconde approche introduite avec le PDP-11, consiste à mapper tous les registres de contrôle dans l'espace mémoire. Chaque registre de contrôle se voit attribuer une adresse mémoire unique à laquelle aucune mémoire n'est assignée. La programmation du contrôleur à partir de l'unité centrale se fait par l'écriture et la lecture de données dans ces adresses. Une des adresses va correspondre à la zone tampon : l'unité centrale y écrira les données à transférer (en sortie) ou y lira les données transférées (en entrée). Une autre adresse correspondra à un registre de commande : l'unité centrale y écrira des valeurs conventionnelles destinées à donner des ordres à l'interface (par exemple exécuter une écriture).

Définition : L'accès direct à la mémoire (DMA)

Un "coupleur DMA" (DMA : Direct Memory Access) (cf. 'Coupleur DMA' p 9) est un type particulier de contrôleur qui est capable de transférer des données directement de la mémoire a un périphérique (ou inversement) sans que ces données ne transitent par l'unité centrale. Donc l'utilisation du DMA décharge l'unité centrale d'une partie importante du travail de contrôle et d'exécution des E/S. La programmation d'un coupleur DMA se fait, comme pour un contrôleur par l'écriture de données dans des "registres". Cette fois il en faut au moins trois :

- l'adresse du début de la zone de mémoire à transférer,
- la longueur de la zone de mémoire à transférer,
- le mot de commande/état.



Coupleur DMA

Définition : Canal d'E/S

Un canal est un petit processeur qui fonctionne comme un dispositif de DMA partagé entre plusieurs interfaces. Le nom "canal" est utilisé surtout parce que c'est le nom qu'avaient reçu ces dispositifs dans les IBM/370 qui ont été des matériels très répandus.

Méthode : Demande d'une E/S

Les instructions d'entrée-sortie sont privilégiées, c'est-à-dire qu'elles ne peuvent figurer que dans des programmes exécutés en mode système. En mode utilisateur un programme demande au système d'opérer l'entrée-sortie dont il a besoin. La seule façon pour un programme d'exécuter une entrée-sortie est de faire un appel à une fonction du système judicieusement choisie par le programmeur. Cette fonction, une fois appelée, va s'exécuter en mode système et aura donc de ce fait accès aux instructions d'entrée/sortie. La forme de cet appel au système est celle d'un appel de procédure.

Il existe dans certains ordinateurs une instruction spéciale SVC ("SuperVisor Call" - appel au superviseur) qui a la forme classique des instructions avec deux paramètres :

SVC n,adresse

Cette forme peut varier suivant les architectures, mais son sens reste celui d'un appel à la routine numéro "n" du système avec en deuxième paramètre l'adresse de la zone contenant ou recevant la donnée à sortir ou à entrer.

Exemple : Demande d'une E/S

On désire faire écrire sur le terminal le contenu d'un tableau de caractères. Si le programme est écrit en langage assembleur, il va faire appel à une primitive d'entrée / sortie du système, par exemple :

```
# mettre sur la pile
MOVL R2, -(SP) # la longueur du tableau
MOVL R1, -(SP) # l'adresse du tableau
MOVL #io_WRITE_BLOC, -(SP) # le code de la fonction à
exécuter
MOVL ICHAN, -(SP)# le numéro de l'unité logique
# appeler la fonction système SYS_qio
# les quatres arguments d'E/S étant rangés sur la pile
CALLS #4, SYS_qio
```

Si ce même programme avait été écrit dans un langage de haut niveau, il aurait pu appeler la même fonction par un ordre du type :

```
status=iowrite(ichan, tab, lng) ;
```

C'est alors le compilateur du langage de haut niveau qui va traduire cet ordre "iowrite" en un appel à la primitive correspondante du système d'exploitation.

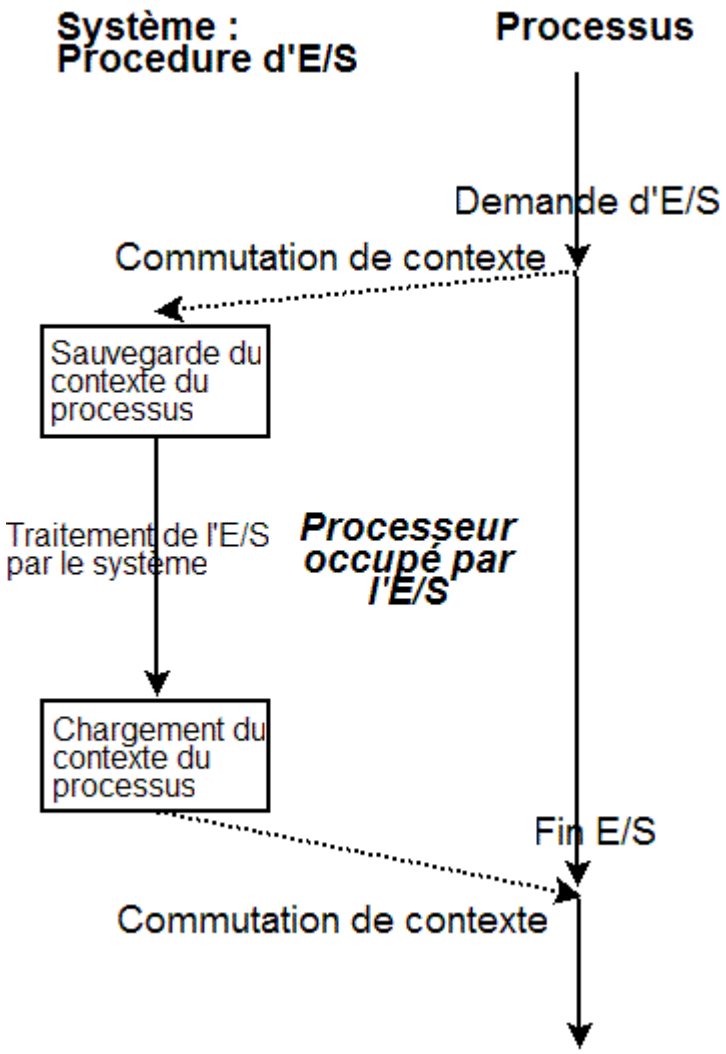
Types d'entrées / sorties

Selon le mécanisme matériel utilisé pour réaliser les entrées / sorties on distingue plusieurs types :

- Les entrées / sorties programmées
- Les entrées / sorties pilotées pas les interruptions
- Les entrées / sorties par DMA
- Les entrées / sorties par canal

Méthode : Les E/S programmées

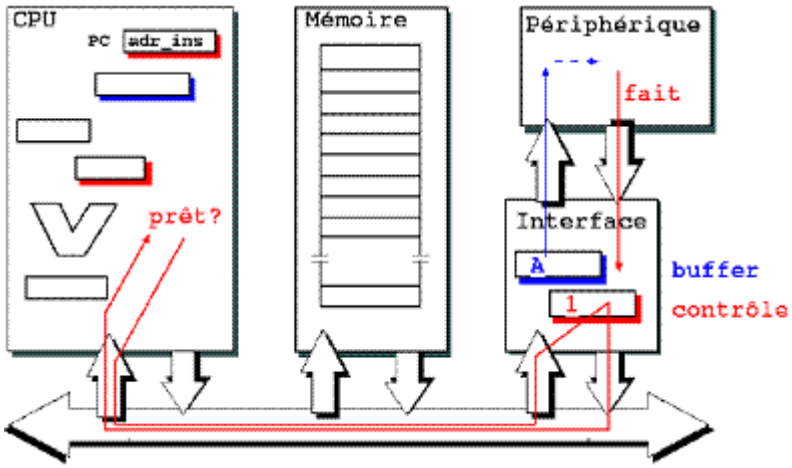
Dans ce cas, le travail est réalisé par le processeur (Figure (cf. 'ES programmées' p 11)).



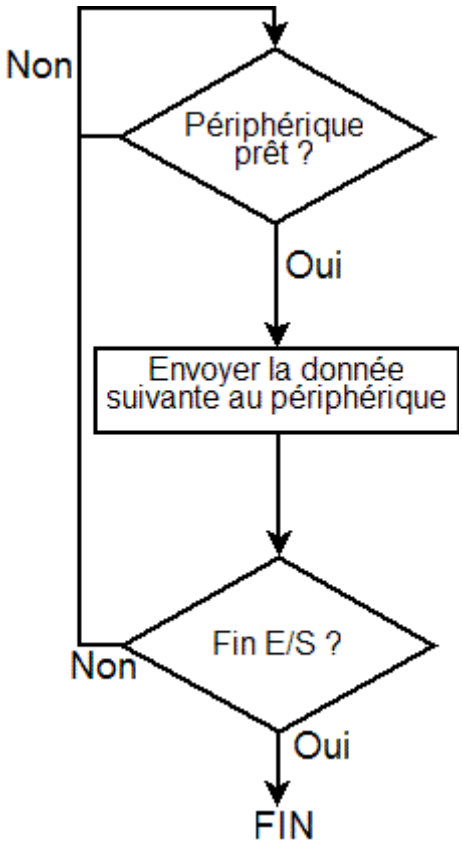
ES programmées

Lorsque le système d'exploitation veut exécuter une opération de sortie, par exemple, il commence par transférer les données dans les registres ou les cellules de la mémoire puis, le système d'exploitation exécute une instruction spéciale qui provoque le transfert vers le périphérique. Cette solution est peu pratique lorsque l'on veut transférer des zones de données d'une taille importante par rapport à ce qui peut être transféré en une fois. En outre, le système doit vérifier la disponibilité du périphérique (polling (cf. 'Busy loop' p 12)) avant de lancer l'opération et doit attendre la fin de cette opération. Il consulte pour cela certains bits du registre d'état du contrôleur du périphérique. Les inconvénients de cette méthode sont importants :

- le CPU est ralenti à la vitesse du périphérique,
- le CPU exécute une boucle d'attente active (busy loop (cf. 'Polling (boucle active)' p 12)) à la place de laquelle il pourrait faire des calculs pour le compte d'autres programmes.



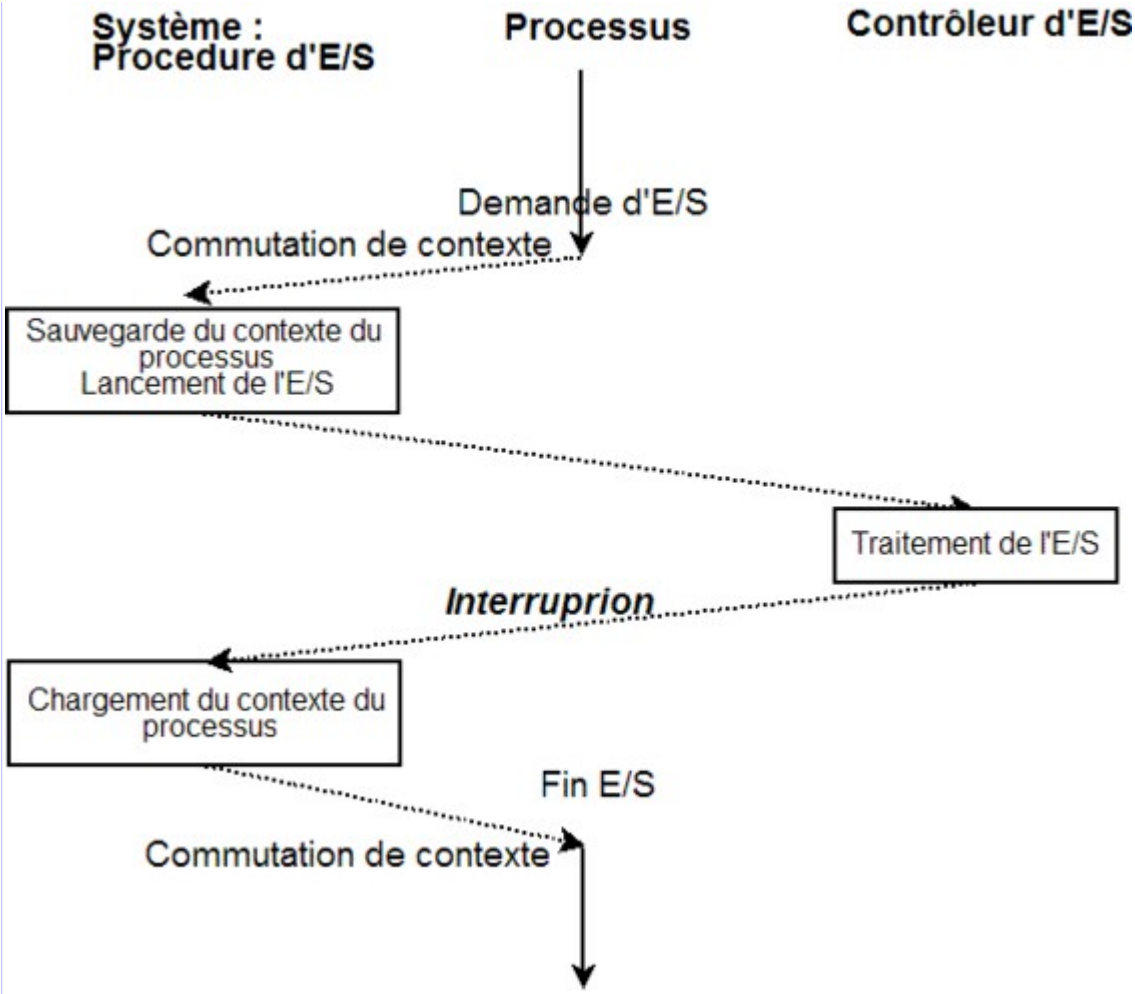
Busy loop



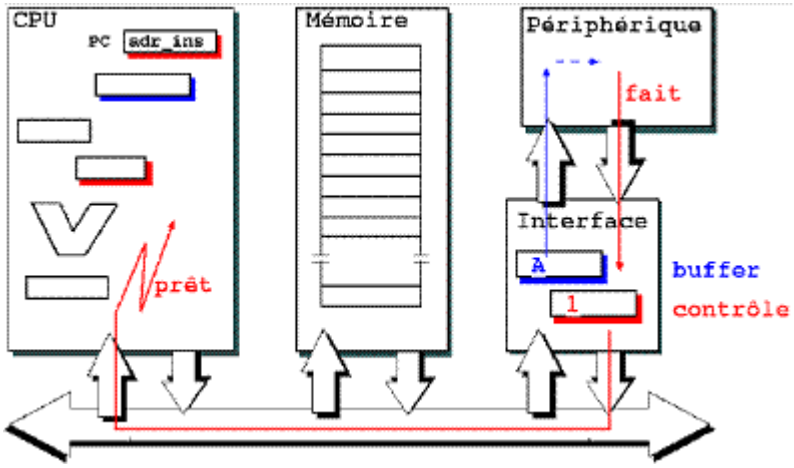
Polling (boucle active)

Méthode : Les E/S pilotées par les interruptions

Dans cette méthode, l'exécution proprement dite de l'entrée sortie est confiée aux contrôleurs d'E/S. Dans ce cas, il faut un moyen pour permettre aux contrôleurs d'E/S de prévenir le système d'exploitation quand une E/S était terminée pour que le système autorise le programme demandeur de l'E/S de poursuivre son exécution. Ce moyen est la notion d'interruption (Figure (cf. 'Pilote d'E/S' p 18)).

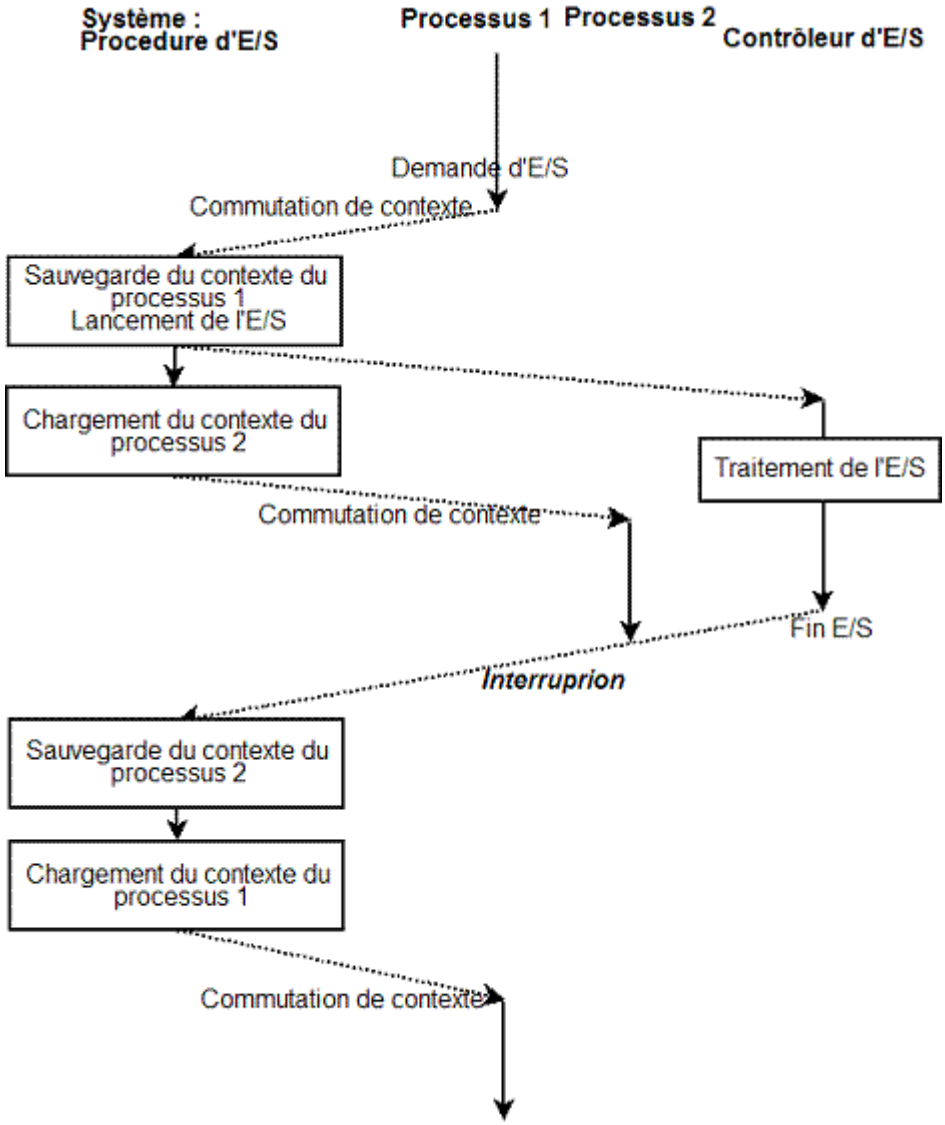


ES pilotées par les interruptions



E/S pilotées par interruptions

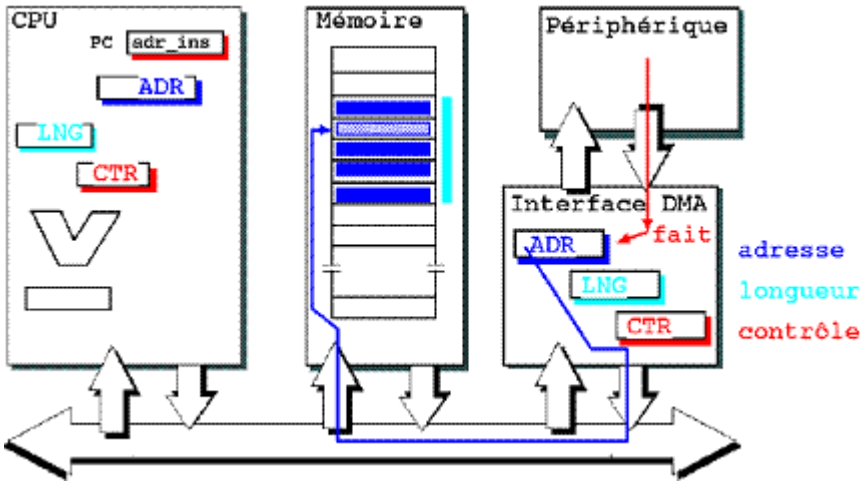
Dans un système temps partagé, le schéma précédent est légèrement différent. En fait, après avoir lancé l'opération d'entrée sortie pour le compte d'un premier processus, le système d'exploitation charge le contexte d'un deuxième processus et lance son exécution. Ceci permet un recouvrement (cf. 'Recouvrement d'ES' p 14) entre le travail du CPU pour le compte d'un processus avec l'activité d'un périphérique travaillant à la réalisation d'une opération d'E/S pour le compte d'un autre processus.



Méthode : Les E/S avec DMA

L'inconvénient évident des E/S pilotées par les interruptions est qu'une interruption se produit à chaque caractère. Comme les interruptions prennent du temps, cette méthode consomme une certaine quantité de temps processeur. Une solution consiste à faire appel au DMA. Dans ce cas, l'idée est de laisser le contrôleur DMA fournir les caractères un à un au périphérique, sans transiter par le processeur central.

Pour l'essentiel, le DMA fonctionne comme une E/S programmée, à la différence que le contrôleur DMA fait le travail à la place du processeur principal. Le grand avantage de l'E/S par DMA (cf. 'E/S par DMA' p 15) est de réduire le nombre d'interruptions. On passe de une par caractère à une par tampon transféré. Si les caractères sont nombreux et les interruptions lentes, l'amélioration peut être considérable *[tanenbaum03]*.



E/S par DMA

Méthode : Les E/S avec canal

Dans les gros ordinateurs, la taille du système et le prix de l'unité centrale justifient que l'on fasse de gros efforts pour optimiser l'utilisation de l'Unité Centrale. La perte de temps inhérente aux E/S programmées rend cette approche pratiquement impossible. L'utilisation d'interfaces communiquant par interruption avec l'UC est aussi difficile pour des raisons d'efficacité (occupation de l'UC pour répondre aux interruptions) et de coût (problème de connexion d'un grand nombre d'interface). De plus, l'efficacité pousse aussi les concepteurs à faire toutes les E/S en DMA pour libérer au maximum une UC très chère. Mais il ne serait pas non plus économique d'équiper toutes les interfaces d'un contrôleur DMA complexe et assez coûteux. D'où l'idée de partager ce contrôleur DMA entre plusieurs interfaces. C'est là l'origine du concept de canal d'E/S. Un canal est donc un processeur, éventuellement spécialisé, qui gère un ensemble de périphériques. A la suite d'une demande d'E/S, le système prépare un programme dans le langage du canal et le met dans la mémoire centrale ou la mémoire du canal. Puis le système lance le canal qui exécutera le programme du transfert proprement dit. A la fin du transfert, le canal envoie une interruption au processeur central qui indique la fin de la transmission.

Un **programme canal** est une suite de commandes qui comportent:

- le nom du périphérique cible,
- le type d'opération (lecture, écriture, avancer bande,...),
- l'adresse d'une zone mémoire,
- la longueur de la zone mémoire.

Bien que les canaux travaillent en parallèle avec le processeur central, il y a de la **concurrence** entre eux pour accéder à la mémoire centrale. En effet, les canaux doivent lire et écrire dans la même mémoire que celle utilisée par l'unité centrale pour l'exécution du programme en cours. Les canaux doivent pouvoir "voler" des cycles d'accès mémoire au processeur central.

C. Gestion de la base de données d'E/S

Plusieurs E/S en papallèle

Plusieurs entrées / sorties peuvent s'exécuter simultanément. Le système doit prendre en charge la totalité de la gestion des entrées / sorties. Ceci nécessite la gestion d'une base de données pour les entrées /sorties en cours afin de stocker et

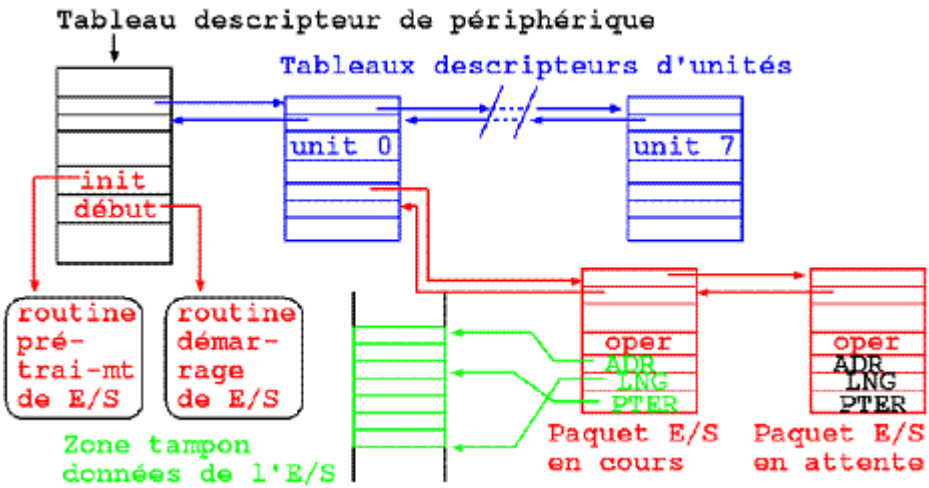
de retrouver facilement les paramètres relatifs à une entrée / sortie lorsque l'unité centrale reçoit une interruption.

Méthode : Base de données des entrées / sorties

A chaque type de périphérique (physique ou virtuel) est associé une table en mémoire (cf. 'Base de données des entrées / sorties' p 16) (dans l'espace réservé au système), appelée "descripteur de périphérique", qui décrit les caractéristiques du périphérique.

Si celui-ci possède plusieurs unités (par exemple un contrôleur de terminaux possédant 8 lignes), à chaque unité est associée une table en mémoire appelée "descripteur d'unité". Chaque entrée/sortie demandée au système par un programme est transformée en un paquet d'informations stocké dans une zone tampon allouée dans l'espace réservé au système. Ce paquet est mis dans une queue des entrées / sorties en attente d'exécution. Ce paquet va contenir des informations telles que :

- adresse en mémoire d'une zone tampon,
- longueur de l'entrée / sortie,
- type de l'entrée / sortie (lecture, écriture, ...)
- adresse dans le programme d'un mot d'état destiné à recevoir le statut final d'exécution de l'opération
- numéro d'un drapeau que le système positionnera à la fin de l'opération (event-flag),
- en option, adresse d'une routine fournie par le programme que le système exécutera à la fin de l'opération



Base de données des entrées / sorties

D. Entrée / Sortie asynchrone

E/S Asynchrone

Les E/S standard sont asynchrones, c'est-à-dire le système ne "rend la main" au programme qu'une fois l'entrée/sortie terminée.

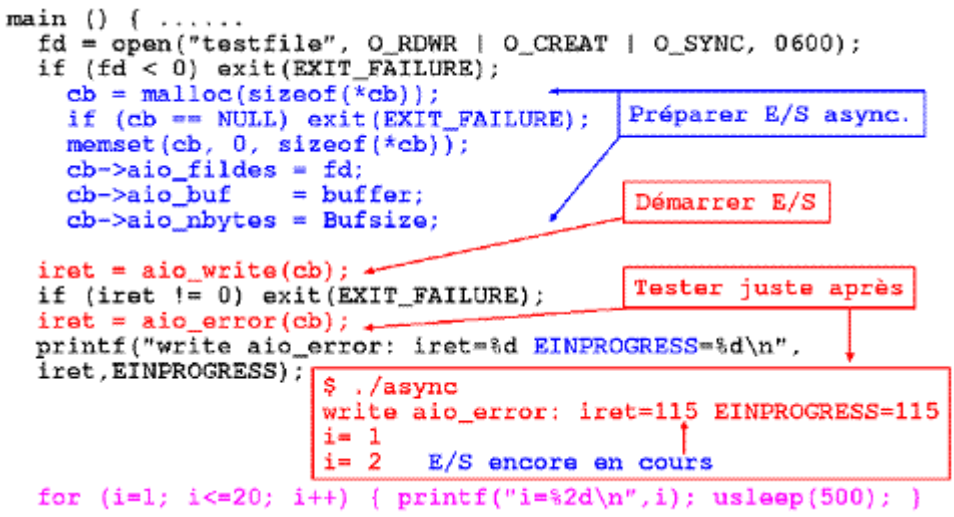
Néanmoins, il est possible pour un programme de demander au système d'exécuter l'entrée/sortie de façon asynchrone : c'est-à-dire de lancer l'entrée/sortie, puis de redonner tout de suite la main au programme.

Le programme peut alors exécuter des calculs "en parallèle" avec le déroulement de

l'entrée/sortie.

Exemple : E/S Asynchrone

On montre ci-dessous un exemple de programmation d'une E/S asynchrone (cf. 'E/S Asynchrone' p 17) :



E/S Asynchrone

E. Pilotes d'E/S

Définition : Pilote

Chaque périphérique d'E/S connecté à un ordinateur doit disposer d'un programme spécifique pour le contrôler. C'est ce programme qui adresse des commandes au périphérique et consulte son état à travers les registres du périphérique. Ce programme est appelé **pilote de périphérique** (*device driver*).

Nom logique / adresse physique de périphérique

Les programmes exécutent leurs entrées / sorties en indiquant au système un "nom logique". Le système maintient une table interne de correspondance entre les noms logiques et les numéros des périphériques.

Le sous-système d'entrée/sortie dirige une requête vers le pilote de périphérique en lui indiquant le numéro du périphérique. Ce programme est le seul dans le système à connaître l'emplacement physique (l'adresse) du périphérique dans la machine.

De plus, seul le pilote connaît des particularités du périphérique telles que les adresses des mots de contrôle, les bits à positionner pour effectuer chaque opération, ...

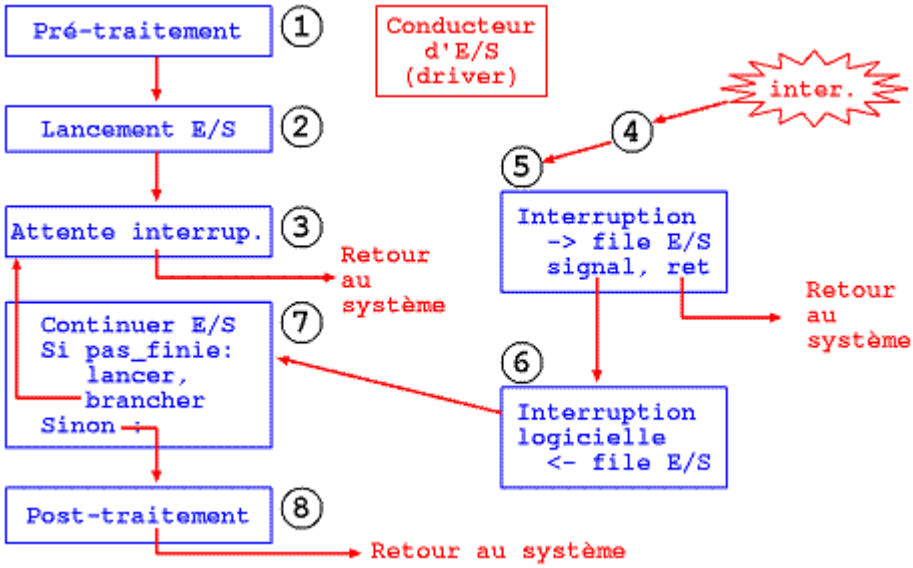
Routines d'un pilote

Un pilote de périphérique (cf. 'Pilote d'E/S' p 18) est constitué de 3 ou 4 routines qui s'appellent les unes les autres selon une séquence précise :

1. Une routine de **pré-traitement** qui vérifie que l'opération demandée est possible (par exemple qu'on ne fait pas une lecture sur une imprimante). Si la demande est correcte cette routine appelle la routine de lancement.
2. Une routine de **lancement** appelée au début de l'entrée / sortie qui est chargée d'initialiser correctement l'interface et de commander le premier transfert. Après cette commande cette routine effectue un branchement sur

la routine d'attente.

3. Une routine d'**attente** des entrées / sorties. Cette routine sauvegarde l'adresse de la routine de continuation dans la base de données des entrées / sorties (dans le paquet associé à l'entrée / sortie en cours). Puis elle rend la main au système.
4. Une routine de **continuation** de l'entrée / sortie en cours. Elle sera appelée par la routine de sortie de file pour poursuivre l'entrée / sortie. Si l'entrée sortie est finie elle appelle une routine de post-traitement du système. Sinon elle lance l'opération suivante et se branche sur la routine d'attente.
5. Une routine d'**interruption**. Elle est exécutée par le système lorsque le périphérique contrôlé par ce pilote envoie une interruption. Elle appelle une routine de mise en file, qui enregistre l'interruption pour un traitement complet ultérieur. Puis elle rend la main au système pour ne pas bloquer les autres interruptions qui pourraient survenir.



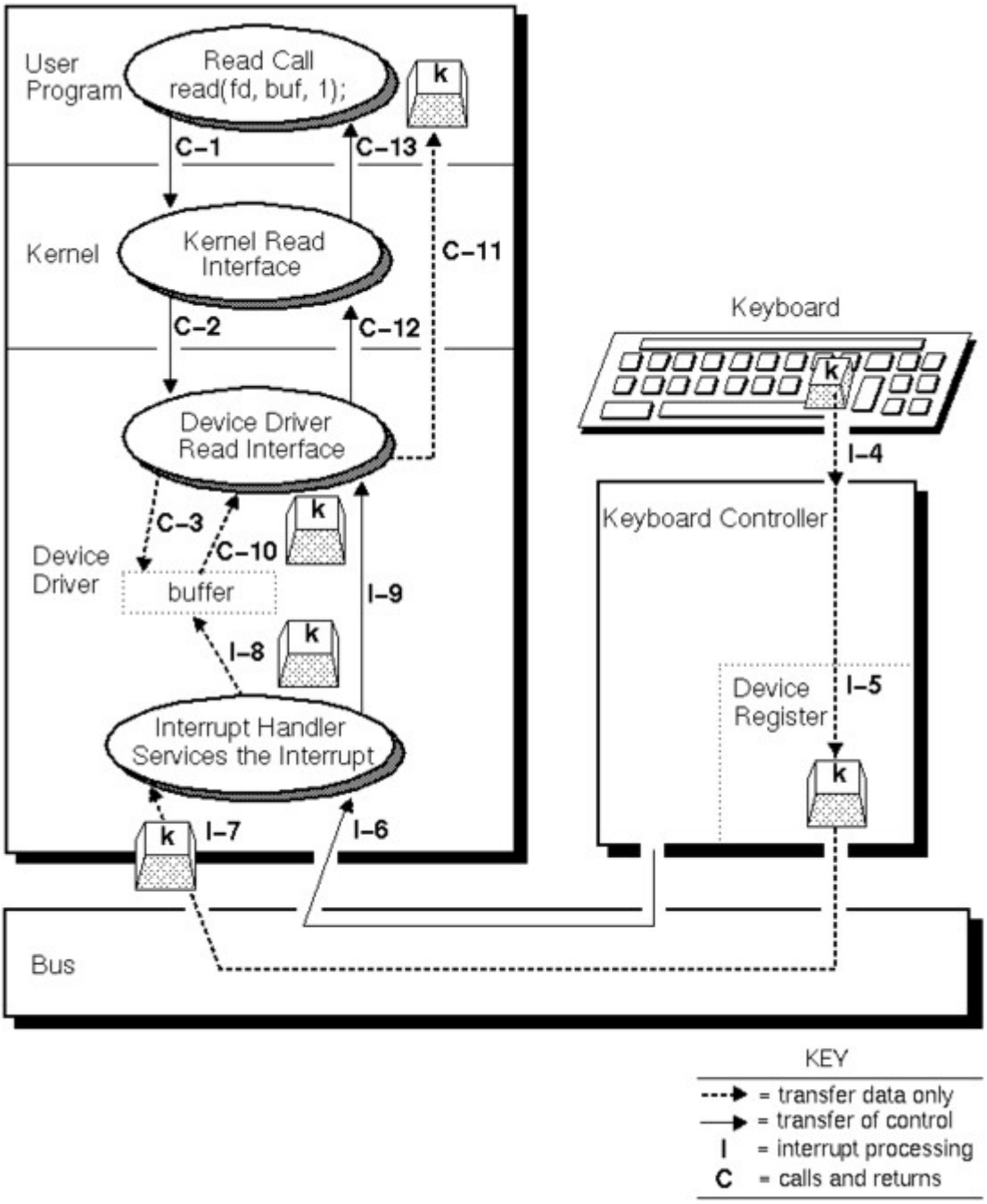
Pilote d'E/S

Hiérarchie des travaux d'entrée / sortie

Le travail à effectuer à chaque interruption est découpé en plusieurs niveaux de priorité décroissante :

1. d'abord répondre aux interruptions,
2. puis continuer les E/S en cours pour ne pas laisser les périphériques inactifs,
3. puis s'occuper de la gestion des E/S terminées,
4. puis, enfin, exécuter les programmes utilisateurs.

Exemple : Lecture d'un caractère [digitalUNIX]



Lecture d'un caractère