ECOLE SUPERIEURE D'INGENIEUR DE RENNES

Université de Rennes

SYS-S5 Les entrées sorties 12/12

Florian Cliquet

Note Taking of Online Resources

9 juin 2024

Plagiarism Mention

We attest that the content of this document is original and stems from our personal reflections.

Sommaire

Introduction

This document provides an overview of SYS-S5 concepts written by Cliquet Florian. It's a set of notes on multiple online resources.

1 Les entrées sorties

Le système d'exploitation contrôle tous les périphériques d'entrées-sorties :

— Il émet les commandes vers les périphériques

— Il intercepte les interruptions

— Il gère les interruptions

Il fournit également une interface simple et facile à utiliser entre les périphériques et le reste du système.

1.1 Aspects matériels des E/S

Les périphériques d'E/S peuvent être divisés en deux catégories en point de vue des informations manipulées :

1. **PES par blocs :**
   Un PES par blocs stocke les informations dans des blocs de taille fixe, chaque bloc possédant sa propre adresse. Il est capable de lire et d'écrire chaque bloc indépendamment des autres.
   `Exemples: HDD, CD-ROM, etc...`
2. **PES de carctères :**
   Un PES de caractère reçoit ou fournit un flot de caractères, sans aucune structure de de blocs. Il n'est pas adressable et ne possède aucune function de recherche.
   Exemples : Imprimantes, souris, etc...

Il est à noter que certains périphériques n'appartiennent à aucune des deux catégories citées comme l'horloge.
Un des aspects matériels qui représente un grand défis aux logiciels gérant les périphériques est la grande différence des débits de transfert de données :

Périphérique	Débit de données
Clavier	10 octets/s
Souris	100 octets/s
Numériseur	400 Ko/s
Caméscope numérique	3,5 Mo/s
Réseau sans fil 802.11g	6,75 Mo/s
CD-ROM 52x	7,8 Mo/s
Ethernet rapide	12,5 Mo/s
Carte mémoire « Compact Flash »	40 Mo/s
FireWire (IEEE 1394)	50 Mo/s
USB 2.0	60 Mo/s
Bus PCI	528 Mo/s
SCSI Ultra 2	80 Mo/s
Ethernet Gigabit	125 Mo/s
Disque dur SATA	300 Mo/s

FIGURE 1

1.1.1 Le contrôleur de périphérique

Pour permettre une conception modulaire et générique, il est souvent possible de diviser le périphérique d'E/S en deux composants, un mécanique (le périphérique lui-même) et un électronique (contrôleur sur puce ou circuit imprimé).

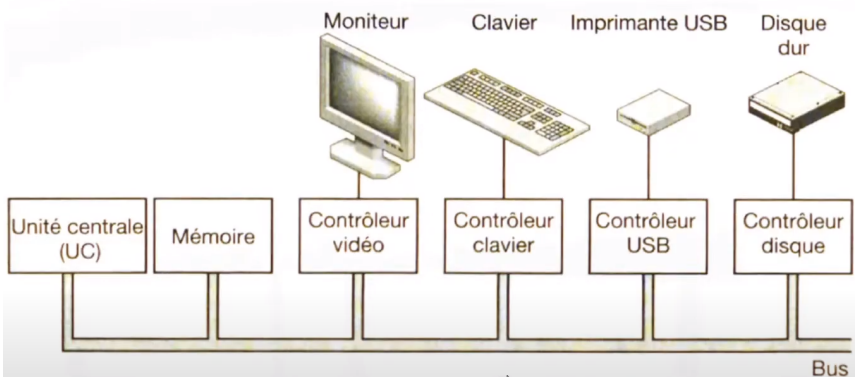

FIGURE 2 – Contrôleur de périphérique

Le contrôleur possède une partie "numérique" pour communiquer avec le proces-

seur et la mémoire et une partie pour envoyer et recevoir des signaux du périphérique. L'interface entre le contrôleur et le périphérique est souvent de très bas niveau.
Le travail d'un contrôleur de disque, par exemple, consiste à convertir le flot binaire série en un bloc d'octets, dans la mémoire tampon du contrôleur,et à apporter toute correction qui s'impose en cas d'erreur. Ensuite, le bloc peut être copié dans la mémoire principale.
Quant à la communication avec le cpu, elle se fait à travers les registres de commandes (contrôles) du contrôleur. En écrivant dans ces registres, le SE demande au périphérique de lui délivrer des données, d'en accepter, de se mettre sous ou hors tension ou encore d'effectuer une action donnée. En lisant ces registres, le SE peut connaître l'état du périphérique, s'il est prêt à accepter une nouvelle commande , etc...
Il existe deux possibilités pour que le cpu communique avec le périphérique :

1. **Port d'E/S :**
   Chaque registre de contrôle se voit assigner un **port d'E/S**, un entier de 8 ou 16 bits. L'ensemble de tous les ports d'E/S forme **l'espace d'E/S** qui est un espace protégé (seul le SE peut y accéder). Les instructions d'E/S utilisées : **IN REG , PORT** ou **OUT PORT, REG**.
2. **E/S mappées en mémoire :**
   Projeter 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.
   Avantages du système d'E/S mappées en mémoire :
   - Manipulation de cases mémoire, ce qui peut être fait dans un langage de haut niveau. Alors qu'avec les ports, on est obligé d'utiliser de l'assembleur pour les IN, OUT, etc...
   - Aucun mécanisme de protection spécial n'est nécessaire pour empêcher les processus utilisateur d'effectuer des E/S. L'espace d'adressage d'un programme utilisateur est différent de celui contenant les registres de contrôle.

1.1.2 L'accès direct à la mémoire (DMA)

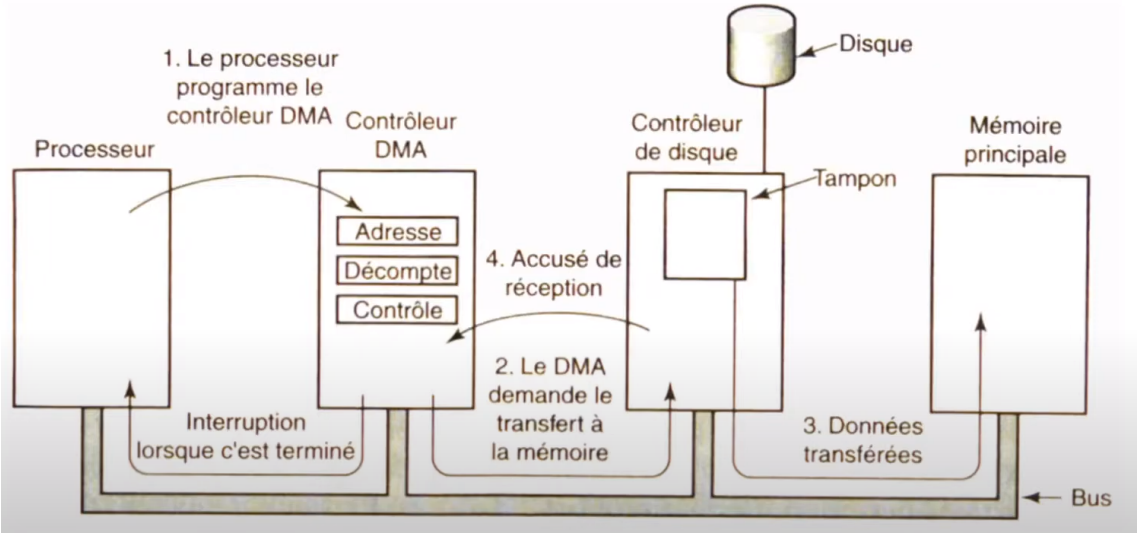

FIGURE 3 – DMA

Un contrôleur DMA peut gérer plusieurs transferts simultanément. Dans ce cas, il doit posséder plusieurs jeux de registres internes.
Le cpu commence par charger chaque jeu de registres avec les paramètres appropriés.
Le DMA peut utiliser un algorithme **à priorité tournante** pour favoriser certains périphériques.
Le transfert peut se faire selon deux modes :

1. **Un seul mot à la fois :**
   Le contrôleur demande le transfert d'un mot et l'obtient. Si le cpu demande lui aussi le bus, il doit attendre (**vol de cycle**).
   Ce mode retarde légèrement le cpu.
2. **bloc :**
   Le contrôleur demande au périphérique d'acquérir le bus, il émet une série de transferts puis libère le bus. Il s'agit d'un fonctionnement en **mode rafale**.
   Plus efficace que le vol de cycle (plusieurs mots pour un acquisition) mais il peut bloquer le cpu et les autres périphériques si une rafale importante est transférée.

1.1.3 Les interruptions

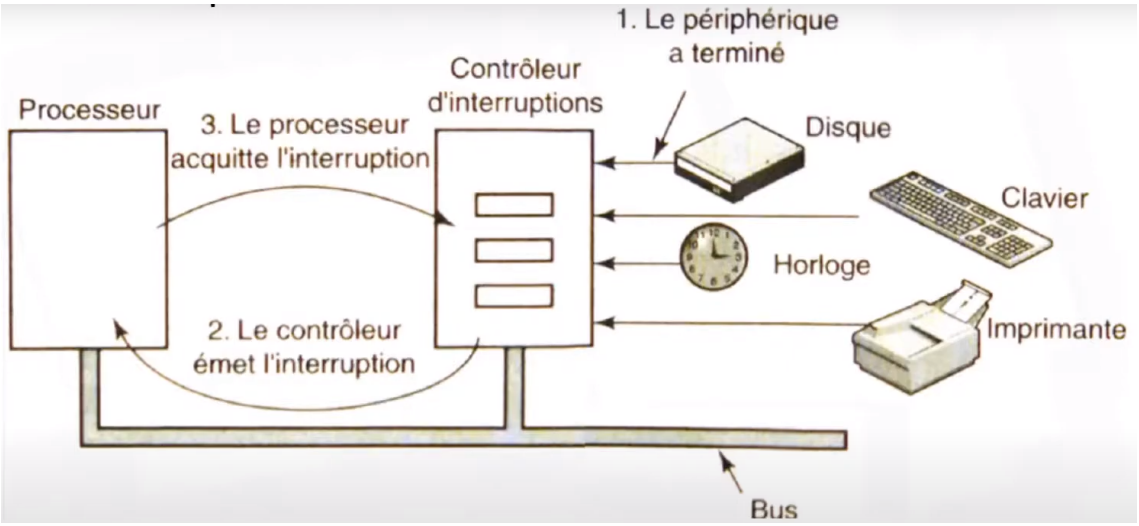

FIGURE 4 – Schéma interruption

Lorsqu'un périphérique termine sa tâche, il émet un signal sur la ligne de bus auquel il a été assigné. La puce du contrôleur d'interruptions détecte le signal et décide de la suite à donner (le traiter ou l'ignorer) et le périphérique continue à émettre le signal jusqu'à ce qu'il reçoive une confirmation de prise en charge du signal par le contrôleur d'interruptions.
Le contrôleur place un numéro sur les lignes d'adressage afin de désigner le périphérique qui réclame de l'attention et émet un signal qui interrompt le processeur.
Le cpu arrête la tâche en cours (sauvegarde du contexte, CO, etc...).
Il utilise le numéro sur les lignes d'adressage comme index dans la table appelée **vecteur d'interruption** pour charger le CO avec une nouvelle valeur et gérer l'interruption.
Les pipelines et l'architecture superscalaires des nouveaux processeurs compliquent la sauvegarde du contexte. Avec ces nouveaux cpu, nous avons des interruptions dites **imprécises ->** les instructions avant le CO ne sont pas tout à fait exécutées et celles après le CO ne sont pas tout à fait non exécutées.

1.2 Aspects logiciels des E/S

Les principes des logiciels d'E/S :

— **Indépendance par rapport au matériel :** Les programmes accèdent à n'importe quel périphérique d'E/S sans préciser le matériel à l'avance

— **Désignation universelle :** Le nom d'un fichier ou d'un périphérique doit être une chaîne de caractères ou un simple entier et ne doit dépendre en aucune manière du périphérique

— **Gestion des erreurs :** Les erreurs sont gérées aussi près du matériel que possible

— La gestion des transferts **synchrones** (bloquants) et des transferts **asynchrones** (pilotés par les interruptions)

— La **mise en mémoire tampon :** souvent, les données qui proviennent d'un périphérique ne peuvent être stockées directement dans leur emplacement final

— La gestion des périphériques d'E/S **partageables**

La structure en couche des logiciels d'E/S :

Logiciel d'E/S (niveau utilisateur)
Logiciel d'indépendance du matériel (niveau SE)
Pilotes de périphériques
Gestionnaires d'interruptions
Matériel

1.2.1 Gestionnaire d'interruptions

1. Sauvegarder tous les registres qui n'ont pas encore été sauvegardés par le matériel d'interruption
2. Configurer un contexte pour la procédure de service de l'interruption (TLB,MMU et table de page)
3. Configurer une pile pour la procédure de service de l'interruption
4. Acquitter l'interruption auprès du contrôleur d'interruptions (autoriser à nouveau les interruptions)
5. Copier les registres depuis l'emplacement où ils sont enregistrés vers la table des processus
6. Exécuter la procédure de service de l'interruption
7. Choisir le prochain processus à exécuter. Si l'interruption a débloqué un processus à haute priorité qui était bloqué, on peut choisir de l'exécuter maintenant
8. Configurer le contexte de la MMU pour le prochain processus à exécuter
9. Charger les registres du nouveau processus
10. Exécuter le nouveau processus

1.2.2 Les pilotes de périphériques

Chaque périphérique d'E/S doit disposer d'un programme spécifique, **pilote de périphérique**, pour le contrôler. Le pilote est généralement écrit, pour chaque SE, par le fabricant du périphérique et livré avec.
Pour accéder au périphérique, registres du contrôleur, la majorité des SEs intègrent le pilote au noyau du SE.

Les SE prévoient une intégration des pilotes dans leur noyau selon l'architecture suivante :

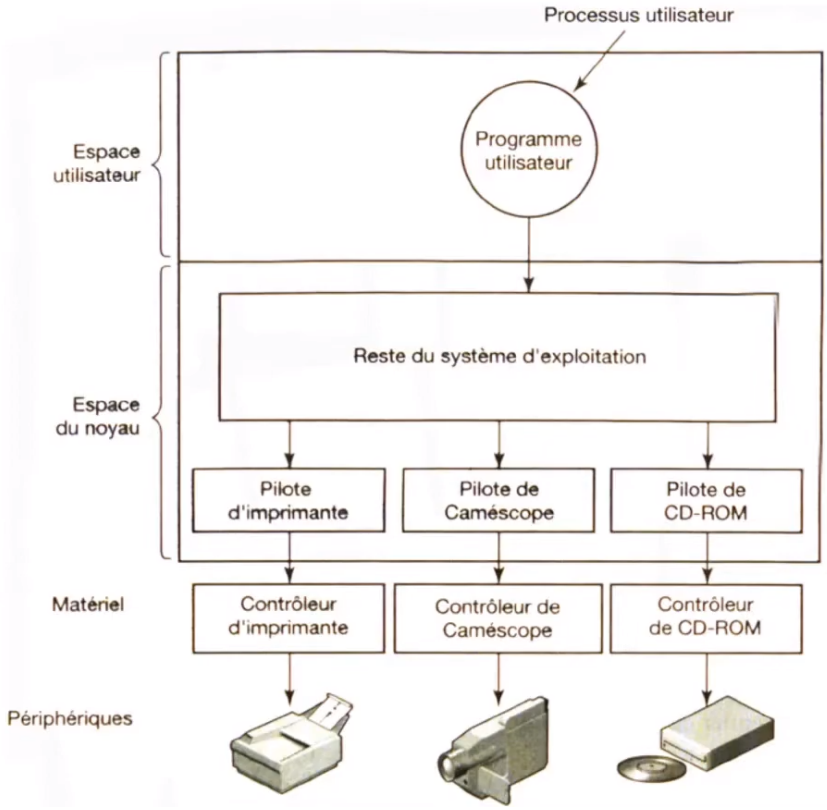

FIGURE 5 – Architecture

— Le pilote commence par vérifier que les paramètres d'entrées sont valides. S'ils ne le sont pas il retourne une erreur.
— Sinon, si nécessaire, il convertit les termes abstraits en termes concrets. (num de bloc à num de tête, piste, secteur et cylindre pour le pilote de disque)
— Il vérifie ensuite si le périphérique est utilisé. Dans ce cas, la requête est placée en file d'attente
— Sinon, il examine le registre d'état du périphérique pour vérifier que la requête peut être gérée
— Il se peut d'allumer le périphérique ou activer une partie matérielle avant de commencer le transfère.

Une fois prêt, le contrôle réel débute :

— le pilote détermine la séquence des commandes à exécuter, qu'il écrit dans les registres du contrôleur
— Après l'écriture de chaque commande, le pilote vérifie que le contrôleur l'accepte et qu'il est prêt à accepter la suivante

— Le pilote peut se bloquer ou non en attendant que le contrôleur termine sa tâche

— Le pilote retourne à son appelant des informations d'état pour le rapport d'erreurs

— Le pilote exécute la prochaine requête ou se bloque en l'attendant

Les pilotes doivent être **réentrants** : un pilote en cours d'exécution doti s'attendre à être appelé une seconde fois avant la fin du premier appel.
Dans un système connectable à chaud (hot pluggable), tandis que le pilote est occupé avec un périphérique, le système peut l'informer qu'il a été retiré. Le transfert doit donc être abandonné sans endommager les structures de données du noyau.

1.2.3 Le logiciel d'indépendance par rapport au matériel

La fonction de base du logiciel d'indépendance par rapport au matériel consiste à assurer des fonctions d'E/S communes à tous les périphériques et à fournir une interface uniforme au logiciel utilisateur.

1. **Interface uniforme pour les pilotes de périphériques**
2. **La gestion des erreurs**
3. **Allocation et libération des périphériques dédiés**
4. **Fournir une taille de bloc indépendante du périphérique**

Interface uniforme pour les pilotes de périphériques :
Il est presque impossible de modifier le SE pour s'adapter à chaque interface d'un pilote. En revanche, il est plus simple d'obliger tous les pilotes à présenter une même interface, cela aide aussi les développeurs des pilotes car ils savent ce qu'on attend d'eux.
Pratiquement, le SE définie un ensemble de fonctions que le pilote doit apporter, le pilote contient donc une table avec des pointeurs vers lui-même pour ces fonctions. Le SE appel ces fonctions d'une façon indirecte via cette table.
La gestion des erreurs :
Les erreurs sont bien plus courantes dans le contexte des E/S que dans d'autres contexte. De nombreuses erreurs sont spécifiques au périphérique et doivent être gérées par le pilote, mais la strucutre de gestion des erreurs est indépendante du périphérique.
A ce niveau, on rencontre deux types d'erreurs :

1. **Erreurs de programmation :**
   Ecrire sur un PE ou lire d'un PS, un périphérique non valide, etc...
   La gestion de ces erreurs consiste à retourner à l'appelant le code de l'erreur.
2. **Erreurs d'E/S :**
   Ecrire dans un bloc de disque endommagé, lire d'un caméscope éteint,etc...

C'est le pilote qui décide quoi faire, sinon il retourne le problème au logiciel d'indépendance du matériel. Ce dernier peut afficher une boite de dialogue avec des options à l'utilisateur, s'il est disponible, pour recommencer ou annuler. Sinon, échec de l'appel système avec un code d'erreur.

Il existe des erreurs plus grave, comme la destruction du répertoire racine, et dans ce cas la, le système doit afficher un message d'erreur et s'arrêter.

Allocation et libération des périphériques dédiés :

Certains périphériques ne peuvent être exploités que par un seul processus à un instant donné (CD-ROM,etc...).

La façon la plus simple de gérer ces périphériques est d'accepter ou rejeter les requêtes d'utilisation selon si le périphérique est disponible ou pas. On peut également, accepter la requête d'utilisation ou la mettre dans une file d'attente et attendre que le périphérique se libère.

Fournir une taille de bloc indépendante du périphérique :

Les tailles de secteurs varient d'un disque à l'autre.

C'est au logiciel d'indépendance du matériel de masquer cela et de fournir une taille de bloc uniforme aux couches supérieurs. Il peut, par exemple, traiter plusieurs secteurs comme un bloc logique.

1.2.4 Les logiciels d'E/S de l'espace utilisateur

Bien que la grande majorité du logiciel d'E/s soit à l'intérieur du SE, une petite partie est composée de bibliothèques liées avec le programme utilisateur.

Les appels systèmes sont émis par des procédures de bibliothèque.

`Exemple : count = write(fd,tampon,nbytes);`

Ces procédures peuvent, en plus de placer leurs paramètres à l'emplacement approprié pour le système, accomplir un travail réel comme le formatage de chaine, etc...

`Exemple : printf("Le carré de %3d est %6d",i,i*i);`

Il n'y pas que les procédures de bibliothèque, le **spoule**, ou désynchronisation des E/S, est une autre partie des logiciels d'E/S.

Il est plus simple de laisser un utilisateur ouvrir le fichier spécial de l'imprimante, mais s'il ne le ferme pas aucun autre processus ne peut utiliser l'imprimante.

Pour éviter ce problème, un processus spécial appelé **démon** et un répertoire, **répertoire de spoule**, sont utilisés.

Les programmes utilisateurs ajoutent leurs fichiers à imprimer dans le répertoire de spoule et le démon est le seul processus qui les imprime en accédant au fichier spécial de l'imprimante.

1.3 Résumé

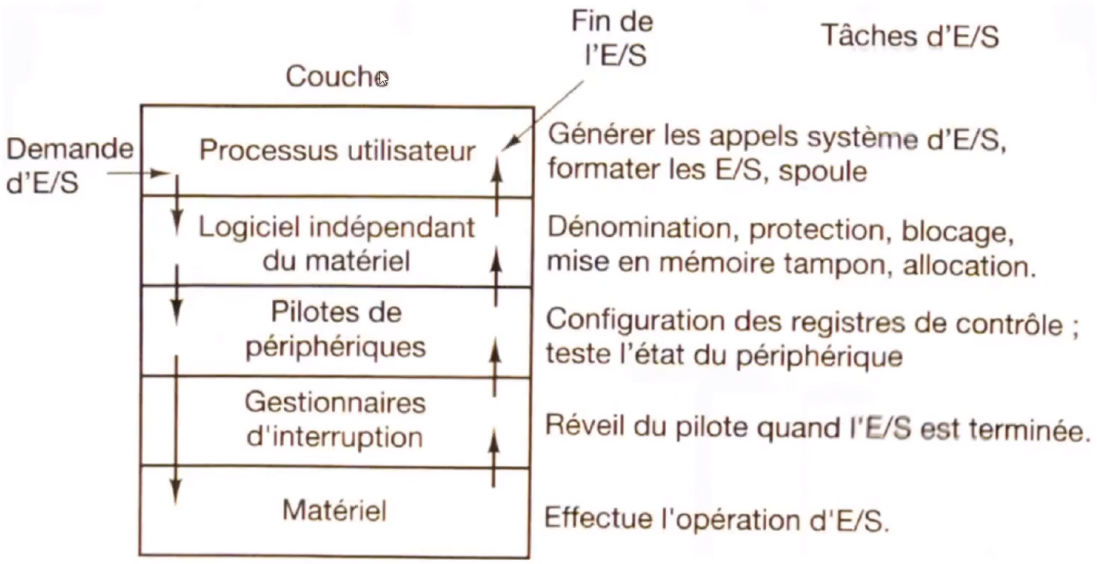

FIGURE 6 – Résumé