SYSTÈME D'EXPLOITATION

M1

Dr MAIOSSOUM Bery

VII
GESTION DES ENTRÉES ET SORTIES

OBJECTIFS

- Les E/S sont le maillon faible du système informatique
  - Périphériques lents
    - Têtes de lecture des disques
- Maximiser l'utilisation pour ne pas réduire l'utilisation processeur
  - Mais plus de processus pour exploiter les processeurs impose plus I/O pour le swap
  - Organiser la vision des périphériques
- Vision uniforme, même si hiérarchique

TYPES DE PÉRIPHÉRIQUES

- **Entrées**
  - Clavier, souris, scanner, micro, capteurs, …
- **Sorties**
  - Ecran, imprimante, …
- **Entrées-sorties**
  - Stockage, réseau, …
- La mémoire n'est pas considérée comme un périphérique
  - Elle est déjà impliquée dans la plupart des instructions.

E/S LOGIQUES OU PHYSIQUES ?

- E/S physiques vers les périphériques
  - Uniquement par l'OS
  - Sauf cas particuliers
- E/S logiques depuis l'application
- Structuration, protection et abstraction assurée par l'OS entre les deux
- Le SE contrôle tous les organes d'E/S : il en récupère les interruptions et fournit les fonctions d'accès.

ASPECTS MATÉRIELS

Contrôleur de périphérique

- On le pilote par ses registres internes :
  - commandes
  - états
  - données

DMA

- Pour faciliter les transferts d'informations avec les périphériques et soulager le CPU, on utilise le DMA (Direct Memory Access).
- Il s'agit d'un dispositif physique auquel le CPU peut demander de faire un transfert entre registres de données du contrôleur de périphériques et mémoire dans un sens ou dans l'autre

ASPECTS MATÉRIELS

- C'est alors le DMA qui règle sa vitesse sur celle du périphérique et génère une IT vers le CPU quand le transfert est terminé
- Un contrôleur de DMA offre plusieurs canaux (une dizaine) càd qu'il peut faire plusieurs transferts en même temps avec plusieurs périphériques.

ASPECTS MATÉRIELS

- **Les interruptions**

- Le fonctionnement est le suivant :
  - génération d'un signal physique sur une ligne spéciale partagée par tous les contrôleurs
  - émission d'un n° d'interruption sur le bus de données (le n° dépend du contrôleur).
- Le SE lui a assigné ce n° lors de l'initialisation du contrôleur pour éviter les numéros en double.

ASPECTS MATÉRIELS

Quand le CPU prend en compte l'IT, il :

- suspend le processus en cours et sauvegarde son contexte
- récupère l'@ où aller dans une table de vecteurs d'ITs en utilisant le n° d'IT comme index
- exécute la procédure prévue qui acquitte l'IT auprès du contrôleur

- Quand c'est terminé, il reprend le contexte du processus interrompu et le continue sauf si l'action associée à cette IT provoque un autre comportement (par exemple lors d'une IT de fin de quantum de temps).

ASPECTS LOGICIELS

- Le principe de fonctionnement est asynchrone :

1. Un processus P demande un transfert (par exemple une lecture sur disque) => appelle une primitive du SE (read)
2. Cette primitive programme le DMA et le contrôleur de périphérique pour le transfert
3. Elle met le processus en état bloqué
4. L'ordonnanceur cherche un autre processus prêt et l'exécute
5. L'ordonnancement des processus se poursuit comme d'habitude
6. L'IT de fin de transfert arrive

ASPECTS LOGICIELS

7. Le processus en cours P' est suspendu et l'IT est traitée
8. Ce traitement d'IT consiste en particulier à rendre le processus P prêt
9. Le processus suspendu P' reprend
10. Plus tard l'ordonnanceur trouvera le processus P prêt, il le rendra actif et P pourra se poursuivre

NB : vu du processus le transfert est bloquant mais vu du CPU il ne l'est pas.

COMMENT LE SE GÈRE LES E/S ?

Le SE gère le E/S en mettant en jeu 4 niveaux :

1. Le gestionnaire d'interruptions

- aspect matériel : prise en compte de l'IT par n° et table d'adresses

2. Les pilotes de périphériques

- gestion du périphérique et de ses ITs : inclus ou ajoutés au SE

3. La partie non tributaire du périphérique

- vue du périphérique par exemple une clé USB est vue comme un disque mais un téléphone peut être vu soit comme un disque soit comme une machine (débogage)

4. Les fonctions offertes aux utilisateurs

- utilisation du périphérique

LE GESTIONNAIRE D'INTERRUPTIONS

La prise en compte d'une IT est relativement complexe car il faut :

1. Sauvegarder les registres du CPU dans la pile (programme interrompu)

NB : On ne peut pas sauvegarder les registres dans la pile du processus en cours car on pourrait tomber sur un défaut de page => une nouvelle IT pour charger la page. Alors, on utilise la pile du SE qui est dans une page fixe toujours en mémoire (exclue de l'allocation de mémoire).

2. Enregistrer le contexte du processus actif dans la table des processus

3. Charger les tables de mémoire virtuelle pour la procédure d'IT

LE GESTIONNAIRE D'INTERRUPTIONS

4. Trouver (table + n° d'IT) et lancer la procédure d'IT
5. Acquitter l'IT vis-à-vis du contrôleur de périphériques
6. Traiter l'IT : code contenu dans le pilote du périphérique

Conclusion : les SE classiques ne sont pas faits pour traiter des ITs très rapidement => non temps réel

LES PILOTES DE PÉRIPHÉRIQUES

- Chaque périphérique a un programme associé : le pilote (driver) éventuellement livré avec le périphérique et adapté au SE
- Un pilote peut contrôler plusieurs périphériques (plusieurs disques par exemple)
- Le pilote fait partie du noyau du SE
- Les SE définissent une interface standard (propre au SE) c'est à dire un ensemble de fonctions que le pilote doit fournir au SE
- Le pilote est chargé dynamiquement par le SE quand il démarre
- Il est indispensable que le pilote corresponde au périphérique ET au système d'exploitation

LES PILOTES DE PÉRIPHÉRIQUES

Ce que fait le pilote :

- vérifier que les paramètres de la requête soient valides
- si le périphérique est occupé, la requête est mise en attente sinon elle est exécutée
- exécuter la requête en accédant aux registres du contrôleur de périphérique (souvent il faut plusieurs étapes) + programmer le DMA
- quand la requête est lancée, le pilote se bloque en attendant que le contrôleur ait fini

LES PILOTES DE PÉRIPHÉRIQUES

- quand l'IT de fin le réveille il transmet les états au logiciel de la couche supérieure (non tributaire du périphérique) et regarde s'il y a d'autres requêtes en attente si oui il lance la suivante sinon il se termine
- **NB** : les pilotes doivent être ré-entrants c'est à dire qu'un pilote doit pouvoir traiter une IT alors qu'il faisait autre chose.
- Par exemple quand on retire une clé USB alors que le pilote était en train de lire ou écrire dessus alors stopper la requête en cours et enlever celles en attente.

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

La couche indépendante du périphérique a pour rôle :

- interfaçage uniforme pour tous les pilotes
- mise en mémoire tampon
- rapport d'erreurs
- allocation et libération des périphériques
- fournir une taille d'information indépendante du périphérique.

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

Interfaçage uniforme :

- Les fonctions offertes par le pilote diffèrent d'un pilote à l'autre
- Les fonctions du noyau dont le pilote a besoin diffèrent d'un pilote à l'autre
- Cette couche se charge d'interfacer le pilote au noyau et aux utilisateurs.
- Exemple : les noms de périphériques
  - sous UNIX **/dev/dsk0**
  - sous windows **C:**

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

- Sous UNIX à un nom de périphérique sont associées 2 valeurs :
  - major number : qui sert à localiser le pilote
  - minor number : qui sert à désigner l'unité pour ce pilote.
- Exemple de lignes du répertoire UNIX /dev :

```
brw-r----- 1 root disk 8,  1 aoû 26 2006  sda1
brw-r----- 1 root disk 8,  2 aoû 26 2006  sda2
crw-rw---- 1 root tty  4, 10 aoû 26 2006  tty10
crw-rw---- 1 root tty  4, 11 aoû 26 2006  tty11
```

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

Mise en mémoire tampon (buffering)

- Pour éviter que le processus ne soit réveillé à chaque information lue sur le périphérique on utilise un tampon
  - on ne le réveille que quand le tampon est plein
  - on gère un second tampon pour mettre les infos qui arrivent quand le 1er est plein et avant que l'utilisateur ne l'ait vidé.
- L'utilisation de tampons permet au SE de mettre en place des stratégies adaptées aux périphériques.

EXEMPLE DE STRATÉGIES POUR LES DISQUES

- **Lecture anticipée sur disque :**

1. Un processus demande la lecture d'un enregistrement (secteur) sur disque

2. Le contrôleur de périphérique déplace la tête et attend que l'enregistrement passe

3. Pendant cette attente plutôt que de ne rien faire on peut lire les enregistrements qui passent et les mettre dans un tampon => si plus tard on nous demande un de ces enregistrements on l'aura déjà => réponse immédiate au processus demandeur.

- **Stratégie d'écriture sur disque :**

Parmi les tampons en attente d'écriture sur le disque on peut choisir d'écrire celui qui est le plus accessible c'est à dire qui demande le moins de déplacement de la tête.

EXEMPLE DE STRATÉGIES POUR LES DISQUES

- **Stratégie d'ascenseur :**

Pour optimiser les déplacements des têtes entre les lectures et les écritures on peut les traiter dans l'ordre des pistes croissant puis décroissant => la tête se déplace vers le centre en traitant toutes les lectures et écritures en attente puis repart vers le bord en traitant toutes les lectures et écritures en attente etc.

- **RAID (Redundant Array of Inexpensive Disks) :**
- On répartit les données sur plusieurs disques qui sont vus par les utilisateurs comme un seul.
  - RAID1 : n disques pour répartir les données + n disques en copie en cas de panne
  - RAID 4 et 5 : un disque est utilisé pour le contrôle d'erreurs des autres => en cas de panne on peut reconstituer les informations perdues en utilisant ce contrôle et reconstituer le contenu du disque planté.

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

- **Mise en mémoire tampon (buffering)**
- L'avantage des mémoires tampons c'est que quand un processus veut écrire des données elles sont mises en tampon et le processus peut continuer
  - – de son point de vue les données sont écrites
  - – en réalité elles ne le sont pas (pas encore)
- L'inconvénient des mémoires tampons c'est qu'en cas de problème certaines données ne sont pas réellement écrites.
- **Exemple** : enlever une clé USB brutalement => risque que certains fichiers soient incorrects alors que normalement ils étaient OK (du point de vue de l'utilisateur).
- C'est pour cela qu'il y a des procédures particulières pour :
  - déconnecter un périphérique amovible
  - fermer une session
  - éteindre la machine.

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

- **Rapports d'erreurs**
- Lors d'une erreur on peut faire différentes choses :
  - afficher un message d'erreur à l'utilisateur
  - planter le processus demandeur
  - tenter de refaire l'opération si c'est possible (par ex pour une écriture disque OK mais pour une gravure sur DVD ?)

LA PARTIE NON TRIBUTAIRE DU PÉRIPHÉRIQUE

Allocation et libération des périphériques

- Certains périphériques (graveur par ex) ne peuvent pas être partagés => utiliser des sémaphores pour mettre en attente les processus qui veulent y accéder jusqu'à ce qu'ils se libèrent ou utiliser un spooler (imprimantes)
- **Fournir une taille de données indépendante du périphérique**
- Par exemple la taille réelle des secteurs des supports de stockage est cachée et on fournit une taille de bloc (cluster) identique pour tous (disques, clé USB).

LES FONCTIONS OFFERTES

- Le SE offre des fonctions d'utilisation des périphériques.
  Par exemple en C sous UNIX : printf , scanf , open , write , read ….
- Certaines de fonctions se retrouvent pour plusieurs périphériques
  par exemple read sur un fichier, une socket, un pipe …)
- Quand un périphérique est non partageable, le SE peut proposer un spooleur (spool = Simultaneous Peripheral Operation On Line). C'est un processus toujours présent (démon) qui gère l'accès à ce périphérique (par exemple : imprimante).

UN PÉRIPHÉRIQUE PARTICULIER : L'HORLOGE (TIMER)

- L'horloge (timer) est un dispositif matériel, un compteur décrémenté à intervalles réguliers (quartz)
  - Programmable en mode répétitif => une IT chaque fois qu'il arrive à 0 et le compteur est automatiquement rechargé à une valeur contenue dans un registre du contrôleur.
  - Programmable en mode mono coup => une IT quand le compteur arrive à 0 puis il s'arrête jusqu'à ce qu'on le relance en réinitialisant le compteur.
  - Utilisé en mode répétitif (chaque 20 à 30 ms) pour le temps partagé
  - Utilisé en mode mono coup pour des surveillances (watchdog timer). Par exemple pour arrêter les disques au bout d'un certain temps de non utilisation.

UN PÉRIPHÉRIQUE PARTICULIER : L'HORLOGE (TIMER)

- NB : le SE gère aussi des timers logiciels càd des variables qu'il décrémente à
- intervalles réguliers (à chaque IT d'un timer répétitif physique).
- C'est moins précis mais ça permet aux utilisateurs de se définir autant de timers logiciels qu'ils veulent
- Exemples : UNIX propose une fonction alarm qui permet à un processus de recevoir un signal SIGALRM dans un délai donné.
- Et aussi une fonction sleep qui met un processus en sommeil pour un délai donné.

FIN