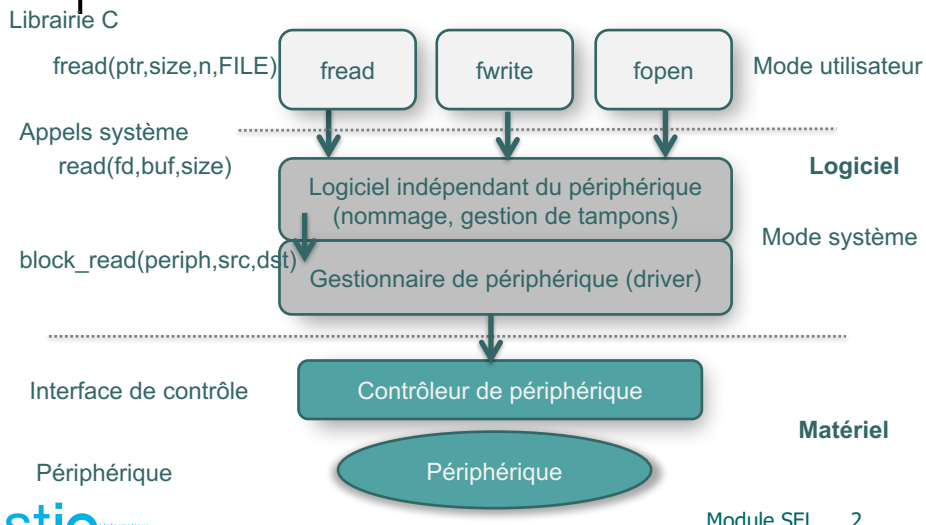
Module SEL

Gestion d'entrées/sorties

- Structure des systèmes d'E/S
- Interface des systèmes d'E/S
- Principes de réalisation : attente active vs interruptions

istic Informatique Électronique

Structure des systèmes d'entrées/sortie



istic Informatique Électronique

Structure générale d'un système d'entrées/sorties

- Eléments mis en œuvre
  - Périphérique : dispositif mécanique, électromagnétique ou électronique assurant physiquement le transfert ou le stockage d'information (disque, clavier, imprimante, etc)
  - Interface de contrôle : circuits assurant la liaison entre le processeur et le périphérique, et constituent le seul moyen d'accès au périphérique
  - Gestionnaire de périphérique (driver) : gère les entrées/sorties physiques en utilisant l'interface de contrôle.

Structure générale d'un système d'entrées/sorties

- Remarques
  - Masquage à l'utilisateur du fonctionnement de l'interface de contrôle et du périphérique
  - Fonctions de bibliothèque : encapsulent les appels système et mettant en œuvre des fonctions supplémentaires (ex: tampons d'entrée/sortie)

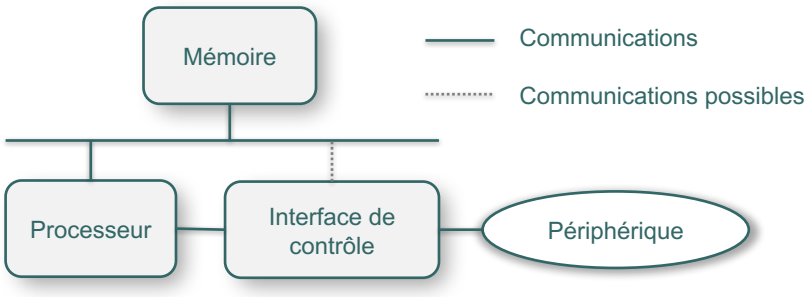
Périphérique (device)

- Types de périphériques :
  - Type bloc : stockage de l'information par blocs de taille fixe, chacun ayant sa propre adresse.
    - Un bloc peut être lu/écrit indépendamment des autres.
    - Exemple : disques, disquettes, CD-rom, etc.
  - Type caractère : accepte ou délivre un flot de caractères sans structure de bloc.
    - Il n'est pas adressable et n'a pas d'opérations de positionnement.
    - Exemple : terminaux, imprimantes, interfaces réseau, souris.
      Essayez ls -l /dev

Interface de contrôle (device controller/adapter)

- Carte insérée dans l'ordinateur, servant à commander le périphérique
- Peut ou non selon les cas transférer directement l'information en mémoire
- Plusieurs degrés de complexité, correspondant à trois grandes classes d'interface de contrôle, de plus en plus sophistiquées
  - Coupleur
  - Coupleur + DMA (Direct Memory Access)
  - Processeur spécialisé (sur bus / réseau)

Interface de contrôle (device controller/adapter)



Entrées/sorties synchrones et asynchrones

- E/S asynchrone (non bloquante)
  - début et fin d'E/S sont vus comme deux événements distincts

```
lancer E/S
...                 Pendant E/S
attendre fin d'E/S
...                 Après E/S
```

- E/S synchrone (bloquante)
  - l'opération d'E/S forme un tout indivisible, le processus ne peut rien faire pendant l'E/S

```
...                 Avant E/S
Faire_E/S
...                 Après E/S
```

Entrées/sorties synchrones et asynchrones

- Le matériel offre des E/S asynchrones. Par défaut, le système d'exploitation offre des fonctions synchrones
- Attention à la présence de tampons, même en synchrone (printf)
  - printf("res = %d\n",res);
  - N'affiche rien, tampon vidé quand plein ou "\n"

Interface

- Interface fichier (open/read/write/close) pour tous type de périphérique
- Périphériques listés dans /dev

Exemple : contrôleur série

- Rôle
  - Transfert d'information (un octet) entre les registres internes du coupleur et le périphérique
- Registres du coupleur vus soit :
  - Comme des adresses banalisées (⇒ manipulées par des instructions de transfert mémoire) : Memory Mapped I/O
  - Dans un espace d'entrées/sorties séparé ⇒ manipulées par des instructions spéciales de type IN ou OUT)
- Le processeur assure lui-même les échanges avec la mémoire

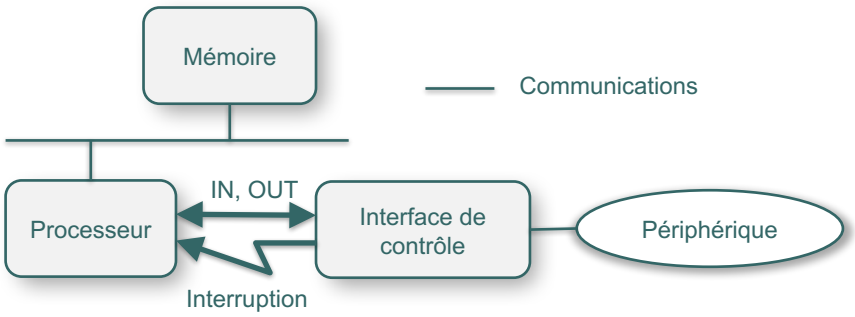
Principe de réalisation

- Synchronisation entre le processeur et le coupleur
  - Adapter le comportement du processeur à l'état du contrôleur (libre, en cours d'E/S)
- Types de mises en œuvre de la synchronisation
  - Par attente active : le processeur lit le registre d'état du contrôleur, jusqu'à ce que celui ci soit dans l'état attendu
  - Par interruption : le contrôleur signale au processeur son changement d'état par une interruption

Matériel considéré : contrôleur série

- Types de registres
  - Registres de données : destinés à contenir les informations échangées avec le périphérique. Il peuvent être lus (entrée) ou écrits (sortie)
  - Registre de contrôle : sert à préciser au coupleur ce qu'il doit faire, et dans quelles conditions (vitesse, format des échanges,...). Ecrit par le processeur
  - Registre d'état : décrit l'état courant du coupleur(libre, en cours de transfert, erreur détectée,...). Lu par le processeur

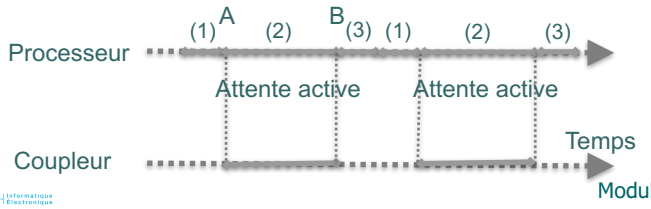
Matériel considéré : contrôleur série



Synchronisation par attente active (test d'état)

- Code (interface de contrôle=coupleur)

```
for (i = 0 ; i < N ; i++) {
        while (controleur.état == occupé) {
                // Attente active
        }
        controleur.donnée = data[i];
}
```



Synchronisation par attente active

- Remarques
  - Le processeur est utilisé pendant l'E/S (attente active)
  - Utilisation inutile du processeur si le périphérique est lent
  - Variation possible : scrutation périodique du registre d'état de l'interface de contrôle (polling)

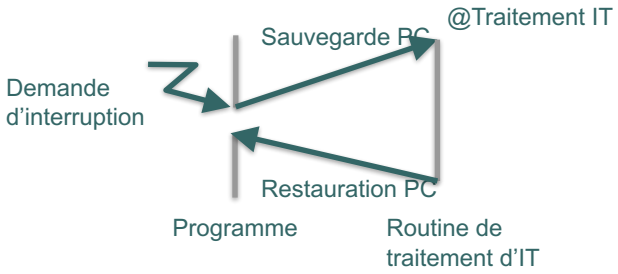
Synchronisation par interruptions

- Principe
  - Le processeur n'attend plus activement la fin de l'E/S
  - Mécanisme d'interruption : prévient via une interruption matérielle la fin de l'E/S

Mécanisme d'interruption

- Lorsqu'un signal d'interruption est émis (demande d'interruption)
  - À la fin de l'instruction en cours d'exécution
  - Sauvegarde de la valeur courante du compteur ordinal (PC)
  - Branchement à une adresse fixe AD_TRAIT_IT
  - Une instruction de retour d'interruption (RTI) affecte PC à la dernière valeur sauvegardée

Mécanisme d'interruption



Mécanisme d'interruption

- Masquage vs suppression d'interruption
  - Suppression d'interruption : configuration du périphérique pour qu'il ne positionne pas de signal d'interruption
  - Masquage d'interruption : configuration du processeur pour qu'il n'effectue pas de déroutement en cas de signal d'interruption (apparait dans le mot d'état du processeur)
    - Instructionx x86 sti/cli

Mécanisme d'interruption

- En général, interruptions masquées pendant l'exécution d'une routine de traitement d'interruptions
  - Sur certaines architectures, priorités entre interruptions
  - Le processeur ne contrôle pas l'instant auquel un signal d'interruption est positionné

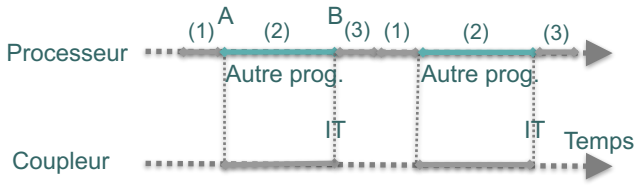
Synchronisation par interruptions

- Une solution possible

```
sema atFin (0);
processus demandeur                    traitement d'IT
for (i = 0 ; i < N ; i++) {            V(atFin)
        controleur.donnée = data[i];   Retour_IT;
        P(atFin)
}
```

Synchronisation par interruptions

- Schéma d'exécution souhaité



Mécanisme d'interruption

- Intérêt
  - Pas de monopolisation du processeur pendant l'attente (exécution d'un autre processus)
- Le schéma d'exécution effectif dépend de la mise en œuvre des interruptions
  - Retour au processus interrompu ou non
  - Priorité des processus
- La routine d'interruption est courte, ce qui est désirable

A retenir

- Pilotes de périphériques : ``Détails'' de fonctionnement du matériel
- Gros impact sur les performances
- Bas niveau, difficiles à mettre au point
- Problèmes de synchronisation subtils
- On a juste touché du doigt les soucis
- Interruptions : permet une bonne utilisation des périphériques et du processeur