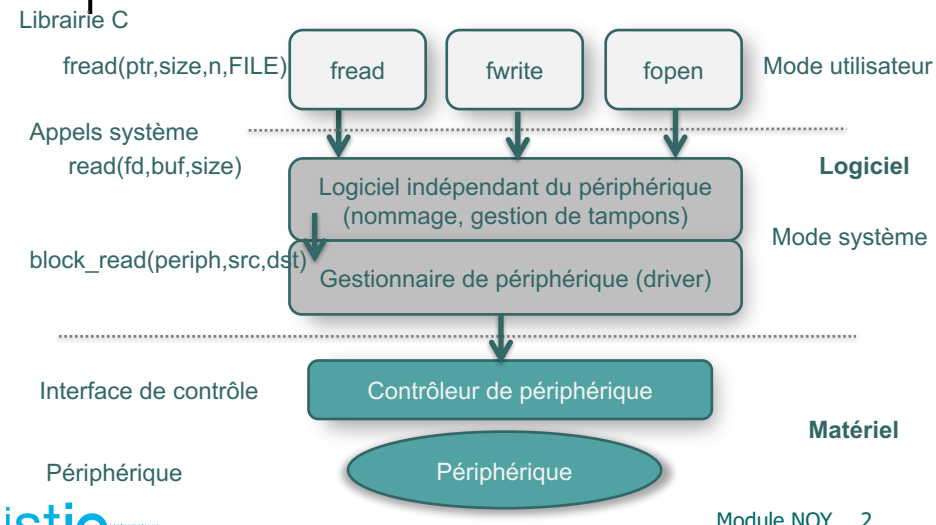
Module NOY

Gestion d'entrées/sorties

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

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)
    - Large gamme de vitesses de transfert
  - Interface de contrôle : circuits assurant la liaison entre le processeur et le 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)

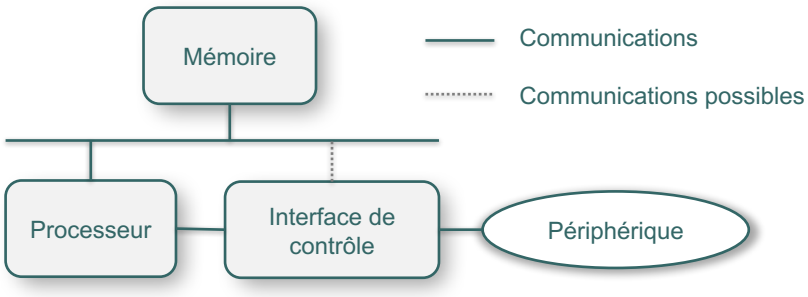
Périphérique (device)

- 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 servant à commander le périphérique
- Peut ou non selon les cas transférer directement l'information en mémoire
- Plusieurs degrés de complexité :
  - Coupleur
  - Coupleur + DMA (Direct Memory Access)
  - Processeur spécialisé (sur bus / réseau)

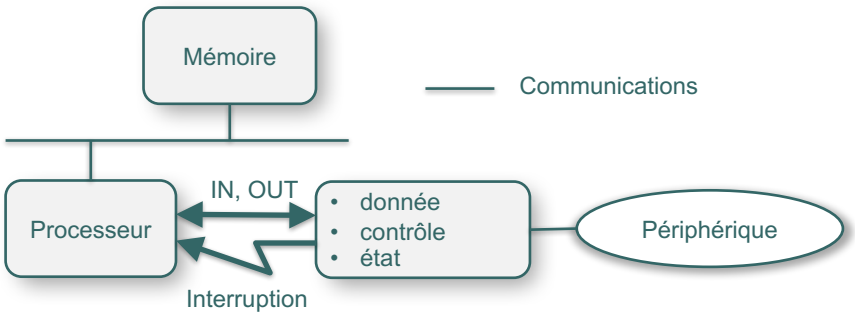
Interface de contrôle (device controller/adapter)



Le matériel : interface de contrôle a. Coupleur

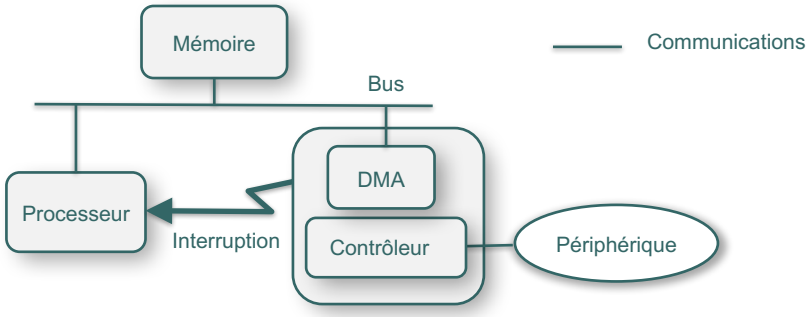
- Rôle
  - Transfert d'information (typiquement un octet) entre les registres internes du coupleur et le périphérique
  - Selon les architectures, registres du coupleur vus
    - Comme des adresses banalisées (⇒ manipulées par des instructions de transfert mémoire) - Memory Mapped I/O
    - Dans un espace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

Le matériel : interface de contrôle
a. Coupleur



Le matériel : interface de contrôle
b. Contrôleur et DMA

- Décharge le processeur des lectures/écritures mémoire
- DMA : circuit qui assure le transfert des informations de l'espace mémoire du contrôleur vers la mémoire



Le matériel : interface de contrôle
b. Contrôleur et DMA

- On indique au DMA une adresse mémoire et un compte d'octets à transférer
- Synchronisation transparente entre le DMA et le contrôleur de périphérique
- Schéma typique du code

```
contrôleur.contrôle = « transfert_avec_DMA »;
DMA.adresse = @mémoire;
DMA.compte = nb_octets;
DMA.contrôle = « interruption »;
// Lancer l'E/S
// Fin transfert signalé dans DMA.état et éventuellement IT
```


Le matériel : interface de contrôle
b. Contrôleur et DMA

- Remarques
  - Processeur déchargé de la gestion des accès mémoire, mais … bus de la machine partagé entre DMA et processeur
  - Modes d'utilisation du DMA
    - Mode cycle stealing : transfert octet par octet, arbitre de bus donne priorité au DMA
    - Mode burst : transfert par bloc, arbitre de bus donne priorité au DMA
    - Mode transparent : arbitre de bus donne priorité au processeur
  - DMA peut aussi être utilisé pour des transferts de mémoire à mémoire

Le matériel : interface de contrôle
c. Processeur d'entrées/sorties

- Rôle
  - Gestion d'E/S complexes de manière totalement autonome, peut gérer plusieurs périphériques
- Domaine d'utilisation
  - Systèmes à forte demande en E/S
  - Périphériques réseau
  - Firmware: logiciel embarqué et exécuté par le processeur d'E/S
- Mise en œuvre
  - File de requêtes en mémoire partagée ou transférées via le réseau
  - Réponses : interruptions ou messages réseau

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"

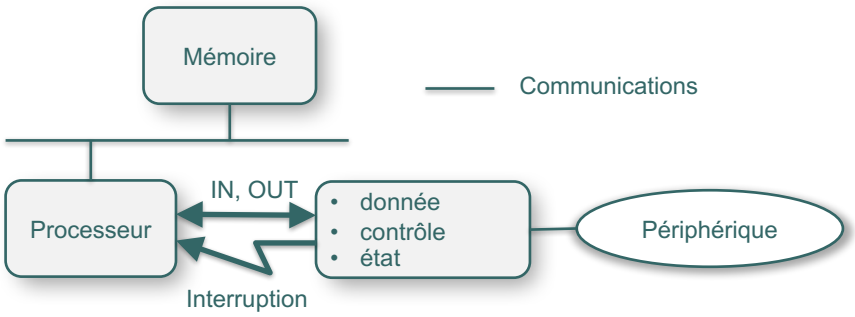
Principe de réalisation

- 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
- Plan de la suite
  - Sur un exemple de contrôleur très simple (coupleur série)

Matériel considéré : coupleur série

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

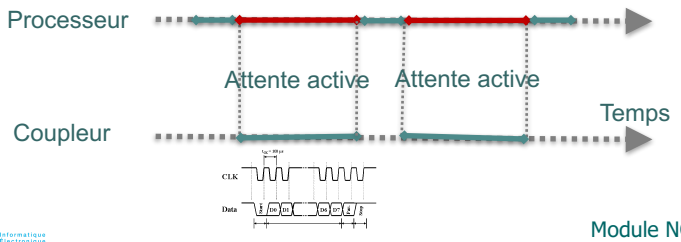
Matériel considéré : coupleur série



Synchronisation par attente active (test d'état)

- Code d'envoi d'une séquence de caractères

```
coupleur.contrôle = « pas d'interruption »; // Initialisation
for (i = 0 ; i < N ; i++) {
      while (coupleur.état == occupé) { ; // Attente active }
      coupleur.donnée = data[i];
}
```



Synchronisation par attente active

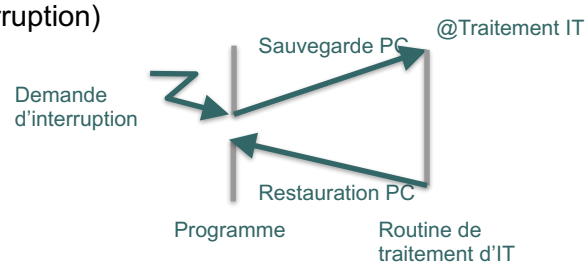
- Remarques
  - Le processeur est utilisé pendant l'E/S (attente active) – phase d'attente non productive
  - Attente active d'autant plus longue que le périphérique est lent
  - Variation possible : scrutation périodique du registre d'état de l'interface de contrôle (polling)

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'entrée sortie
- Intérêt
  - Pas de monopolisation du processeur pendant l'attente (exécution d'un autre processus)
- Remarque
  - Interruption en sur transition ici (fin d'E/S) ici, le code serait différent avec interruption sur état

Mécanisme d'interruption : rappel

- Lorsqu'un signal d'interruption est émis (demande d'interruption)



- Contexte minimal sauvegardé en cas d'interruption (PC,SR) par le matériel
- Une routine d'interruption n'a pas de contexte
  - Pas de blocage possible, pas d'appel système bloquant autorisé

Mécanisme d'interruption : rappel

- Un niveau d'interruption (IT) peut être :
  - Armé/désarmé (activé/désactivé) : suppression de la source de l'interruption (signal d'interruption)
    - Obtenu par configuration du contrôleur de périphérique
  - Masqué/démasqué : configuration du processeur pour qu'il n'effectue pas de déroutement en cas de signal d'interruption
    - Stocké dans le mot d'état du processeur
    - Instructions spécifiques (instructions privilégiées cli/sti)

Mécanisme d'interruption : rappel

- Remarques
  - 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
  - Acquittement des interruptions : lecture registre d'état

Synchronisation par interruptions

- Une solution possible

```
sema s_fin (0);
coupleur.contrôle = « interruption »; // Initialisation
```

Emetteur

```
for (i = 0 ; i < N ; i++) {
   coupleur.donnée = data[i];
   P(s_fin)
}
```

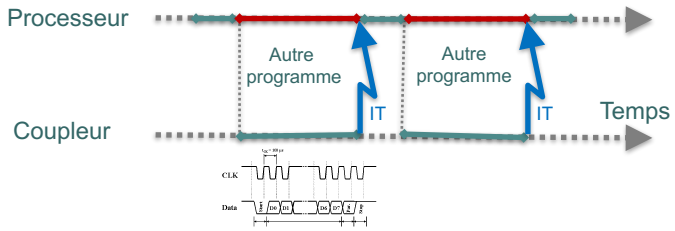
Traitement d'IT

```
V(s_fin)
Retour_IT;
```

- Remarque
  - La routine d'interruption est courte, ce qui est désirable

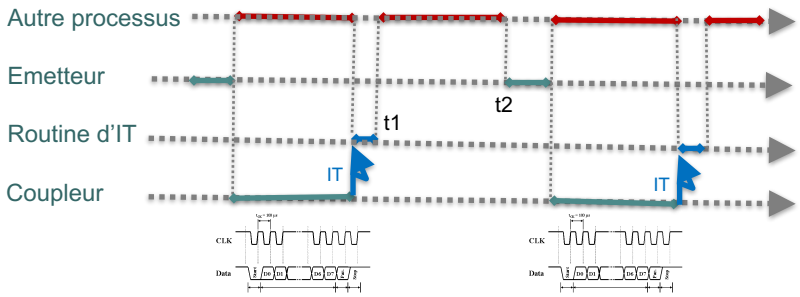
Synchronisation par interruptions

- Schéma d'exécution souhaité



Synchronisation par interruptions

- Schéma d'exécution effectif si retour au processus interrompu



Synchronisation par interruptions

- Problème de la solution présentée
  - Le processus qui a demandé l'entrée/sortie n'est pas relancé immédiatement après (en t2 alors que ce serait possible dès t1)

  ⇒ mauvaise utilisation du périphérique
- Solutions possibles
  - Entretien de l'E/S dans la routine d'IT
  - Ré-ordonnancement en fin de routine de traitement d'IT

Synchronisation par interruptions

- Avec entretien de l'E/S dans la routine d'IT

```
sema s_fin (0);
  int cpt = 0;
  coupleur.contrôle = « interruption »; // Initialisation
```

Emetteur

```
cpt = 0;
coupleur.donnée = data[cpt];
P(s_fin);
```

Traitement d'IT

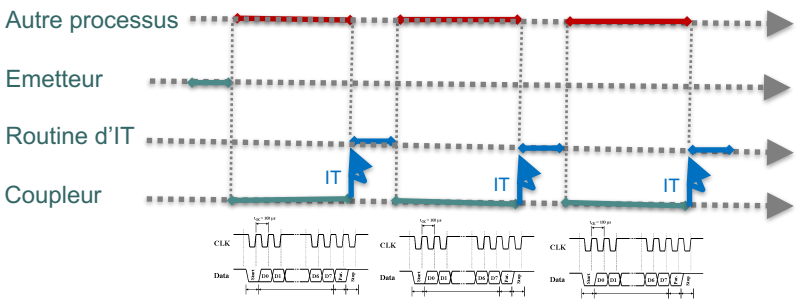
```
if (cpt < N) {
  cpt = cpt+1;
  coupleur.donnée = data[cpt];
} else V(s_fin);
Retour_IT;
```


Synchronisation par interruptions

- Remarques
  - Partage de variable entre émetteur et routine d'interruption. Comment les protéger des accès concurrents ?
    - Sémaphore inadapté : routine d'IT n'est pas un processus ⇒ on ne peut pas faire de P bloquant
    - ⇒ Autres moyens : tryP, ouverture des ITs uniquement après initialisation des variables partagées

Synchronisation par interruptions

- Diagramme temporel

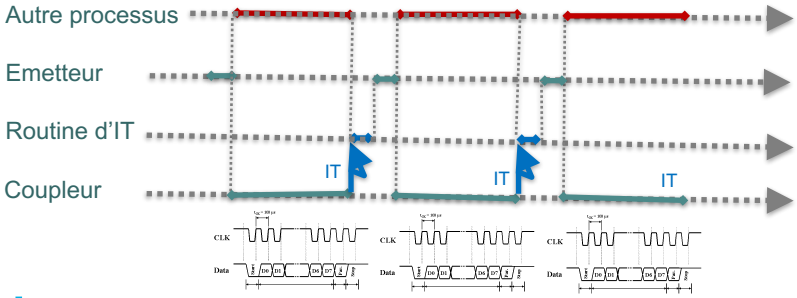


Synchronisation par interruptions

- Remarques
  - Une seule « vraie » commutation de processus
  - Routine d'IT plus complexe
    - Problème potentiel : perte d'IT si le traitement d'IT devient trop long
  - Code plus complexe à écrire (partage de variables entre driver et routine d'IT)

Synchronisation par interruptions

- Schéma d'exécution effectif si :
  - Appel à l'ordonnanceur en fin de routine d'IT
  - Emetteur très prioritaire
  - Code identique à la solution de départ



Synchronisation par interruptions

- Remarques
  - Relance du processus en attente ⇒ meilleure utilisation du périphérique
  - Un changement de contexte par caractère

Gestion d'interruptions dans les systèmes d'exploitation modernes

- Découpage des routines d'IT en deux morceaux
  - First-level interrupt handlers (upper half dans Linux)
    - Rôle : minimum vital pour gérer l'interruption
    - Si besoin, planification d'un second-level interrupt handler pour les traitements les plus longs
  - Second-level interrupt handlers (lower half ou bottom half dans Linux)
    - Rôles : traitements plus longs à effectuer lors d'une interruption, interruption démasquées
    - A un contexte d'exécution propre et est ordonnancée de manière similaire aux threads

Gestion d'interruptions dans les systèmes d'exploitation modernes

- Re-schedule or not to re-schedule?
  - Reschedule
    - Bénéfice : tout appel système dans une routine d'interruption prend effet à la fin de l'exécution de la routine
    - Désavantage : sauvegarde de tout le contexte à chaque IT
  - Réponses
    - Systèmes généralistes, dont Linux, oui
    - Systèmes spécifiques (OSEK/VDX), au choix
      - Deux types d'interruptions au choix du développeur (catégories 1 et 2)

Attente active vs interruptions Discussion

- Attente active
  - Monopolisation du processeur lors de la phase d'attente
  - Mais pas de changement de contexte !
- Interruptions
  - On exécute du code utile lors des E/S
  - Mais changement de contexte, avec le coût (sauvegarde/restauration de registres)

  ⇒ Dans le cas ou le périphérique est rapide, on peut avoir intérêt à utiliser l'attente active, si la durée de la phase d'attente active est inférieure au temps de changement de contexte

Attente active vs interruptions Discussion

- Dans les solutions par IT, les traitements d'IT doivent être brefs (interruptions masquées). Avec un traitement d'IT trop long, on risque de ne pas être apte à traiter d'autres ITs en temps utile

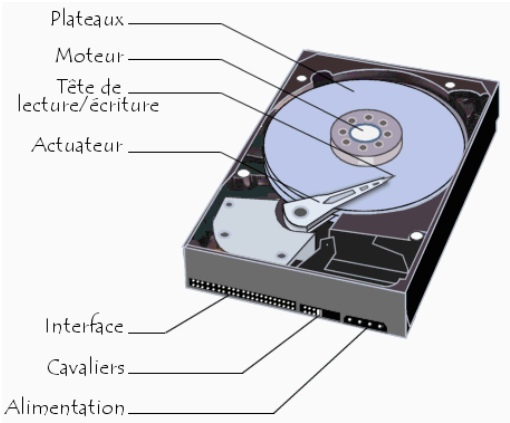
  ⇒ Schéma à privilégier :

  traitement d'IT : V(...)
  Retour_IT

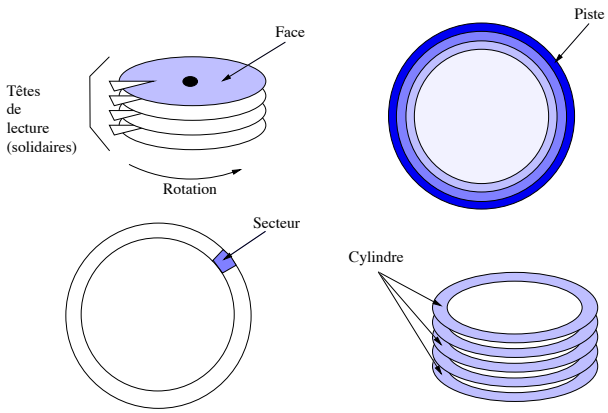
  (ou partie longue dans le bottom half)
- Si plusieurs processus utilisent le même périphérique/coupleur/DMA, il va bien entendu falloir assurer l'exclusion mutuelle sur ces éléments.

Gestion des disques

- Disques



Gestion des disques

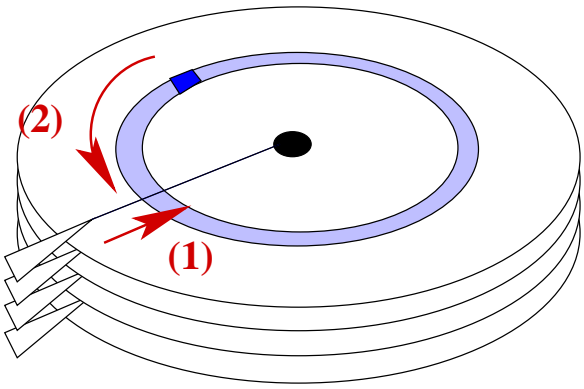


- Zone Bit Recording: plus de secteurs sur pistes externes (densité)

Gestion des disques

- Adresse disque = triplet (num. face, num. piste, num. secteur) (traduction numéro secteur → triplet faite par le contrôleur)
- Décomposition d'une E/S disque
  - Positionnement piste, on déplace le bras sur la piste indiquée (seek time);
  - Latence (rotational latency), on attend que le secteur désigné passe sous la tête (statistiquement, 1/2 tour)
  - Transfert proprement dit d'un ou 2 secteurs consécutifs

Gestion des disques



Gestion des disques

- Remarques
  - Attente du secteur et le transfert lui-même indissociables
  - Le contrôleur disque dispose d'un tampon interne
    - Pour s'affranchir de problèmes temporels pour le recopie en mémoire (suivre le débit du disque)
    - Pour accélérer le traitement des requêtes en évitant des accès disque
  - Seek time variable selon la distance à parcourir

Gestion des disques
Amélioration des temps de réponse

- Eléments considérés
  - Positionnement du bras sur la piste voulue
    - Temps important (ms)
    - Pas de transfert pendant le positionnement
    - Solutions : politiques d'ordonnancement de requêtes
  - Temps de transfert proprement dit
    - Paralléliser des accès
    - Anticiper des accès

Gestion des disques
a. Politiques de gestion du bras

- Exemple
  - Disque de 512 pistes par face (numérotées de 0 à 511 à partir de l'extérieur), 100 secteurs par piste
  - Temps de passage d'une piste à l'autre de 0,5 ms
  - Temps de rotation de 10 ms
  - Durée (moyenne) d'une E/S d'un secteur (ms) = $T_{posit}+T_{transfert}$ =p*0.5 + 5.1, avec p le nombre de pistes à traverser
  - Requêtes de lecture de secteur pour les pistes 300, 6, 200 (dans l'ordre)
  - Initialement, bras positionné sur la piste 0

Gestion des disques
a. Politiques de gestion du bras

- Version 1 : traitement des requêtes dans l'ordre d'arrivée
  - T_{ES1} (0 → 300) = $T_{posit}+T_{transfert}$ = 300*0.5 + 5.1 = 155.1 ms
  - T_{ES2} (300 → 6)= 294*0.5 + 5.1 = 152.1 ms ;
  - T_{ES3} (6 → 200)= 194*0.5 + 5.1 = 102.1 ms ;
  - Total = 409.3 ms
- Version 2 : traitement des requêtes dans 6, 200, 300
  - T_{ES2} (0 → 6) = T_{posit+T_{transfert =6*0.5 + 5.1 = 8.1 ms
  - T_{ES3} (6 → 200)= 194*0.5 + 5.1 = 102.1 ms ;
  - T_{ES1} (200 → 300) = 100*0.5 + 5.1 = 55.1 ms
  - Total = 165.3 ms

Gestion des disques
a. Politiques de gestion du bras

- Temps de réponse pour les requêtes

Requête	Version 1	Version 2
300	155.1	165.3
6	307.2	8.1
200	409.3	110.2

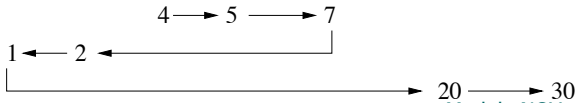
  - Version 2 : le système a mis globalement moins de temps, mais une requête a été servie un peu moins vite
    ⇒ Equilibre à trouver entre deux phénomènes
    - Amélioration du débit global
    - Ne pas trop défavoriser des requêtes particulières (et éviter la famine)
  - Ces politiques n'ont d'intérêt que quand on charge le disque

Gestion des disques
a. Politiques de gestion du bras

- FCFS (First Come First Served) Premier arrivé - premier servi (version 1)
  - Requêtes traitées dans l'ordre du dépôt
  - Traitement équitable des requêtes (absence de famine)
  - Mais beaucoup de déplacements du bras

Gestion des disques
a. Politiques de gestion du bras

- SSTF (Shortest Seek Time First)
  - Plus petit déplacement d'abord : privilégie la requête qui, à partir de la position courante, nécessite le moins de déplacement du bras
  - Exemple : Requêtes concernant les pistes 1, 2, 5, 7, 20, 30, départ de la piste 4, pas d'autres requêtes arrivant par la suite
  - Traitement des requêtes dans l'ordre 5, 7, 2, 1, 20, 30



Gestion des disques
a. Politiques de gestion du bras

- SSTF (Shortest Seek Time First)
  - Diminue le nombre de déplacements du bras (tout en n'étant pas optimale)
  - Non équitable, le temps d'attente d'une requête est non borné (risque de famine)
  - Exemple : nombreuses requêtes, concernant des pistes toujours proches de la position courante de la tête

Gestion des disques
a. Politiques de gestion du bras

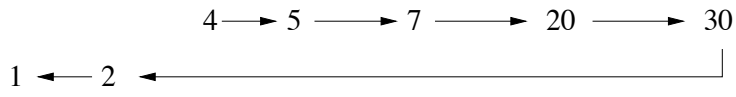
- Politiques "de l'ascenseur »
  - Ensemble de politiques
  - Principe commun
  - Déplace systématiquement du bras d'un coté à l'autre des faces
  - On sert les requêtes concernant une piste quand le bras passe au dessus de cette piste

Gestion des disques
a. Politiques de gestion du bras

- Version la plus simple des politiques « de l'ascenseur » (SCAN)
  - Déplacement du bras d'un bord à l'autre des faces (jusqu'au bout)
  - Transferts dans les deux sens de déplacement du bras

Gestion des disques
a. Politiques de gestion du bras

- Version la plus simple des politiques « de l'ascenseur » (SCAN)
  - Requêtes concernant les pistes 1, 2, 5, 7, 20, 30, départ de la piste 4 en direction de la 511, pas d'autres requêtes arrivant par la suite
  - Traitement des requêtes dans l'ordre 5, 7, 20, 30, 2, 1



Gestion des disques
a. Politiques de gestion du bras

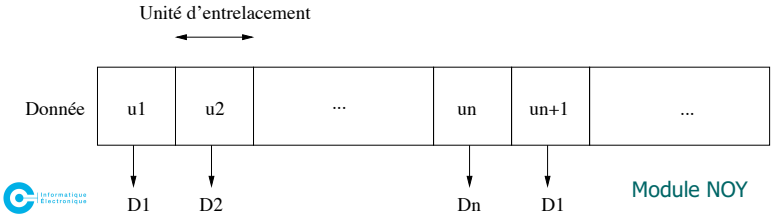
- Evaluation de la politique SCAN
  - Optimise les déplacements du bras
  - Fournit un temps de service borné pour une requête
  - Pas totalement équitable : les requêtes concernant les pistes du milieu sont avantagées

Gestion des disques
a. Politiques de gestion du bras

- Evaluation de la politique SCAN
  - ⇒ Variantes :
    - C-SCAN : on ne fait des transferts que dans un sens de déplacement, retour en un seul coup sans transfert
    - LOOK : on arrête de se déplacer dans un sens quand il n'y a plus de requêtes concernant les pistes entre la position courante et le bord, dans le sens du déplacement, transferts dans les deux sens
    - C-LOOK : même chose que LOOK, mais on ne fait les transferts que dans un sens
- Eviter les déplacements du bras « en amont »
  - Allocation disque intelligente
  - Défragmentation

Gestion des disques
b. Accélération des accès : disk arrays

- Tableau de disque ("disk array") : collection de disques fonctionnant en parallèle et vus comme un disque unique
- Principe : répartir la donnée à échanger sur les différents disques, on parle d'entrelacement
- Exemple : u1, u2 etc. représentent des unités d'entrelacement que l'on répartit sur n disques (sans redondance)



Gestion des disques
b. Accélération des accès : disk arrays

- Taille de l'unité d'entrelacement
  - Octet
    - Augmentation du débit du transfert pour les gros transferts
      - Traitement en parallèle des requête
      - Masquage du délai de positionnement
  - Bloc
    - Répartition automatique de la charge sur les différents disques

Gestion des disques
b. Accélération des accès

- Observations sur les accès disque
  - Accès fréquent de certains secteurs
  - Dans certains cas, possibilité de prédire les accès futurs (accès séquentiels)
  - ⇒ Deux types d'optimisation
    - Caches disque : on conserve en mémoire une copie de l'ensemble des blocs les plus à même d'être réutilisés dans le futur
    - Pré-chargement : on pré-charge en mémoire les blocs qui ont des chances d'être lus dans le futur
  - Différentes localisations pour les caches
    - Mémoire du contrôleur disque, RAM

Fiabilité

- Avec les tableaux de disque, augmentation du nombre de disques ⇒ augmentation de la probabilité d'une défaillance d'un disque
- Utilisation de redondance (duplication de l'information) pour tolérer les pannes de disque (disques RAID - "Redundant Arrays of Independant Disk") à l'origine "Redundant Arrays of Inexpensive Disk »
- Erreurs locales (secteurs défectueux) peuvent être détectés par des codes détecteurs et correcteurs d'erreur sur le secteur
- ⇒ On s'intéresse ici à la défaillance d'un disque complet, qui interdit tout accès à ce disque

Fiabilité

- Méthodes pour tolérer la défaillance complète d'un disque
  - Méthode simple : deux copies de chaque disque (disques miroir)
  - Méthode plus économe : un disque de contrôle C pour deux disques de données D1 et D2C[i] = D1[i] xor D2[i]
  - Défaillance totale du disque D2 ⇒ on peut reconstituer D2 grâce à D1 et C (D1[i] xor C[i])

D1	D2	C=D1 xor D2	D1 xor C
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

Fiabilité

- RAID 0 : pas de redondance, entrelacement (avec une grosse unité d'entrelacement on vise à augmenter le débit en octet/s, avec une petite c'est le débit en requête/s que l'on veut augmenter)
- RAID 1 : redondance par disques en miroir
- RAID 2 : accès en parallèle, petite unité d'entrelacement, code de hamming
- RAID 3 : accès en parallèle, petite unité d'entrelacement, bit de parité ;
- RAID 4 : accès indépendants, grosse unité d'entrelacement, parité sur un disque
- RAID 5 : idem RAID 4, mais les bits de parité sont répartis sur tous les disques

A retenir

- Pilotes de périphériques : « détails » de fonctionnement du matériel
  - (le diable est dans les détails)
- Gros impact sur les performances
- Interruptions
  - Permet une bonne utilisation des périphériques et du processeur
- Bas niveau, difficiles à mettre au point
- Problèmes de synchronisation subtils
- On a juste touché du doigt les problèmes
  - Travail d'expert