le cnam

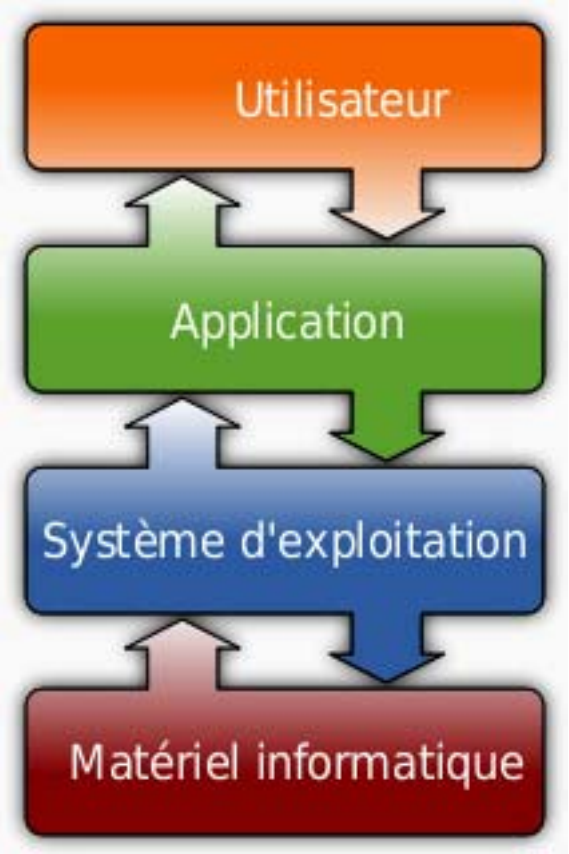
Le système d'exploitation

NSY 104

Introduction

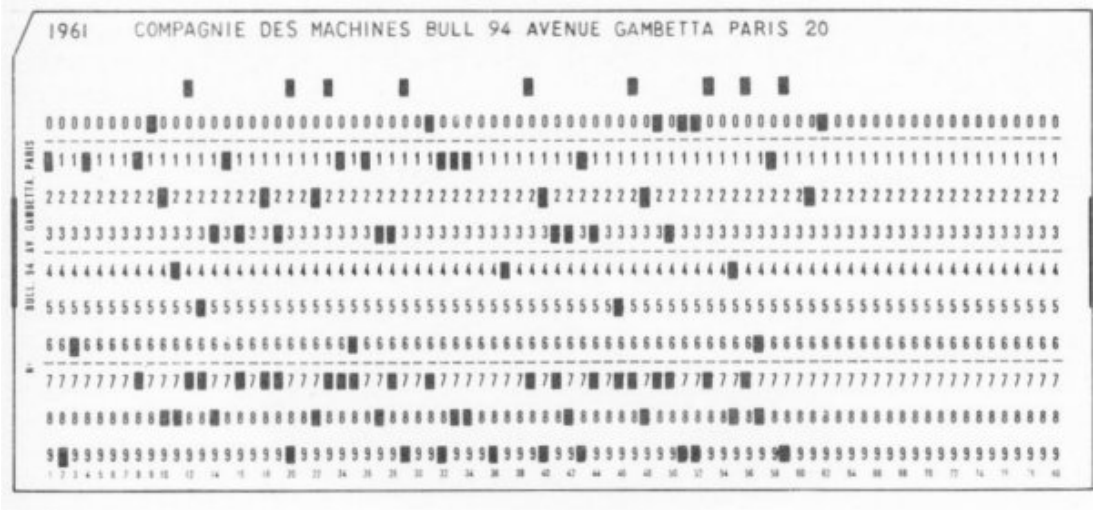
- MACHINE VIRTUELLE (ETENDUE)
  - Permettre aux programmes d'utiliser les composants matériels de façon simple et efficace

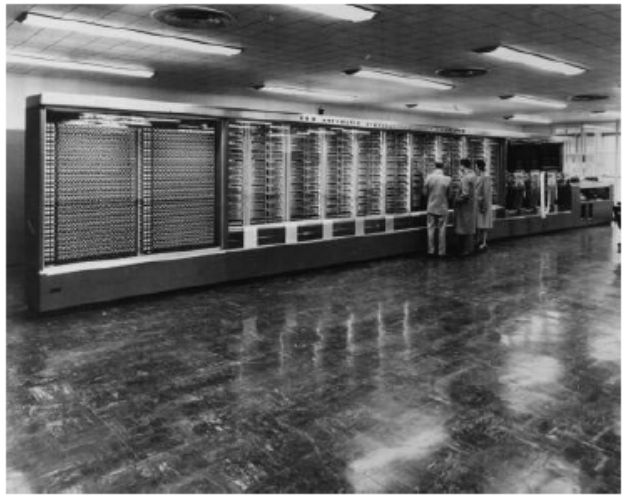
- GESTIONNAIRE DE RESSOURCES
  - Ordonner et contrôler l'allocation des processeurs, des mémoires et des périphériques entre les différents programmes qui les sollicitent.



Historique

- **Première génération, 1945 – 1955 : Tubes à vide**
  - Modification des circuits électriques pour concevoir l'équivalent d'un programme informatique.
  - Réservation de la machine et insertion des cartes modifiés électriquement.
  - A partir de 1950, utilisation de cartes perforées.



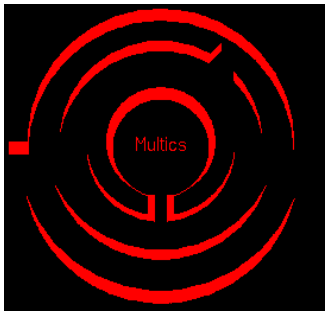


Historique

- **Deuxième génération, 1955 – 1965 : les transistors**
  - Notion de *jobs* : un opérateur prend dans la salle de soumission des travaux un travail sur carte perforées, le charge dans l'ordinateur et met le résultat dans la salle de retraits des travaux: la machine passe son temps à attendre les opérateurs.
  - Notion de traitements par lots : on transfère le contenu de plusieurs travaux sur cartes perforées sur bande magnétique en utilisant de petites machines. Ensuite, on insère les bandes magnétiques sur l'ordinateur principal, les résultats étant délivrés sur bandes magnétiques, on effectue l'opération inverse en convertissant les bandes en cartes perforées.

Historique

- **Troisième génération, 1965-1980: les circuits intégrés**
  - Multiprogrammation et connexion en ligne de plusieurs centaines d'utilisateurs.
  - Naissance et mort de MULTICS (http://web.mit.edu/afs/net/user/srz/www/multics.html).



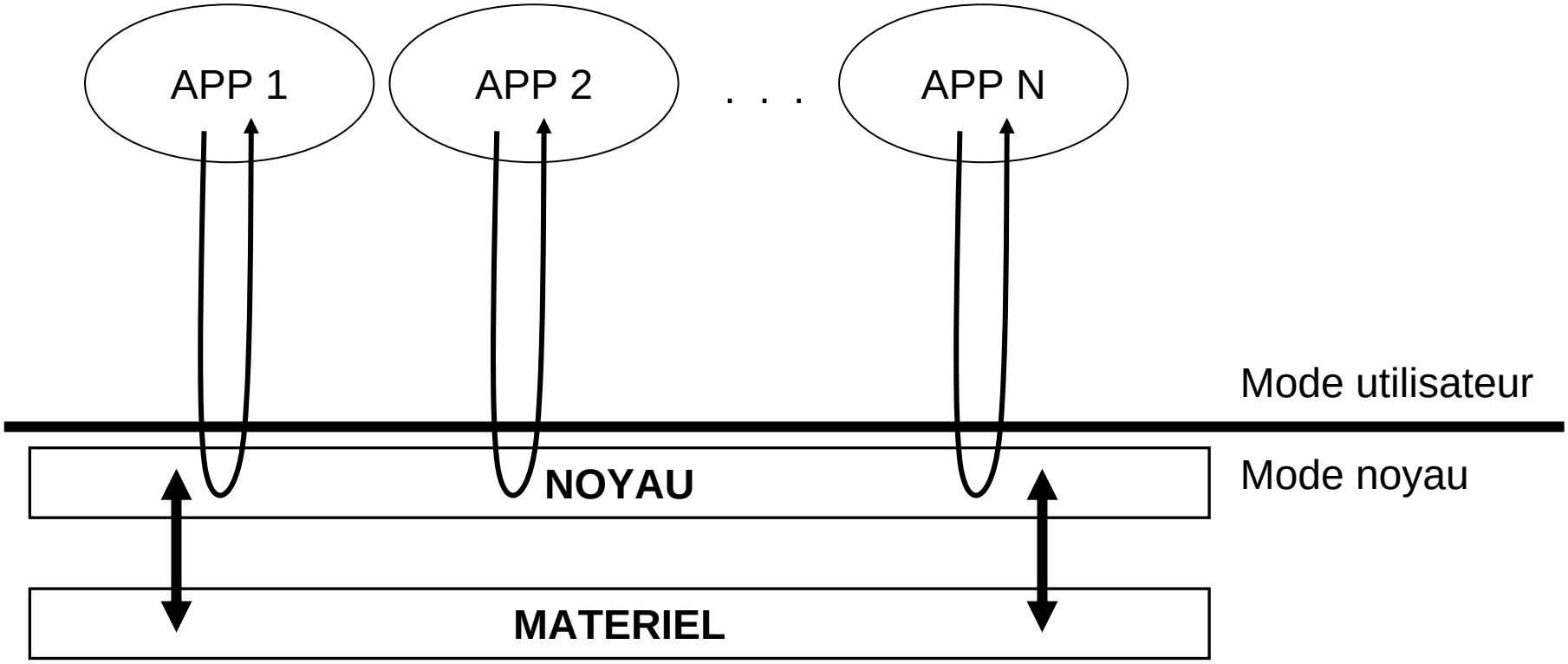
Historique

- **Quatrième génération, 1990-20xx: les ordinateurs personnels**
- Réseau d'ordinateurs :
  - Système d'exploitation intégré « autosuffisant ».
  - Connaissance des autres machines pour échanger des données.
  - Systèmes distribués :
    - Exécution des programmes sur plusieurs processeurs.
    - Déroulement des algorithmes d'ordonnancement même si les informations sont fausses à cause des délais de transmission.

Appel système

- Les programmes communiquent avec le système par l'intermédiaire d'instructions étendues.
  - Ces instructions sont nommées « appels systèmes ».
- Ces appels créent, utilisent et détruisent des objets qui sont gérés par le système.
- L'appel système est une fonction:
  - appelée depuis un programme de l'espace utilisateur
  - dont le traitement est effectué dans l'espace noyau
  - dont le retour est effectué dans le programme appelant dans l'espace utilisateur.
- On y associe systématiquement
  - Un mode d'exécution
  - Des changements de contexte

Modes Noyau / Utilisateur

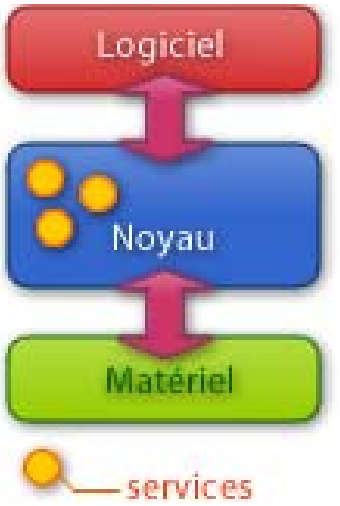


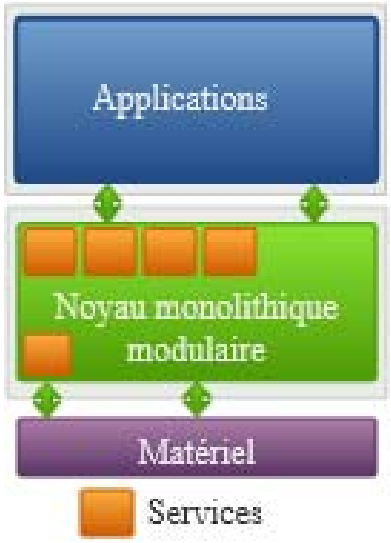
Fonctions du noyau

- Gestion des processus
  - Chargement, exécution
  - Processus légers (threads)
- Gestion des E/S
  - la gestion des systèmes de fichiers ;
  - des supports réseaux (TCP/IP, pare-feu, ...)
  - des services réseau (NFS, ...)
- Pilotes pour le matériel
- Fonctionnalité dans le noyau = performances +++

Architectures

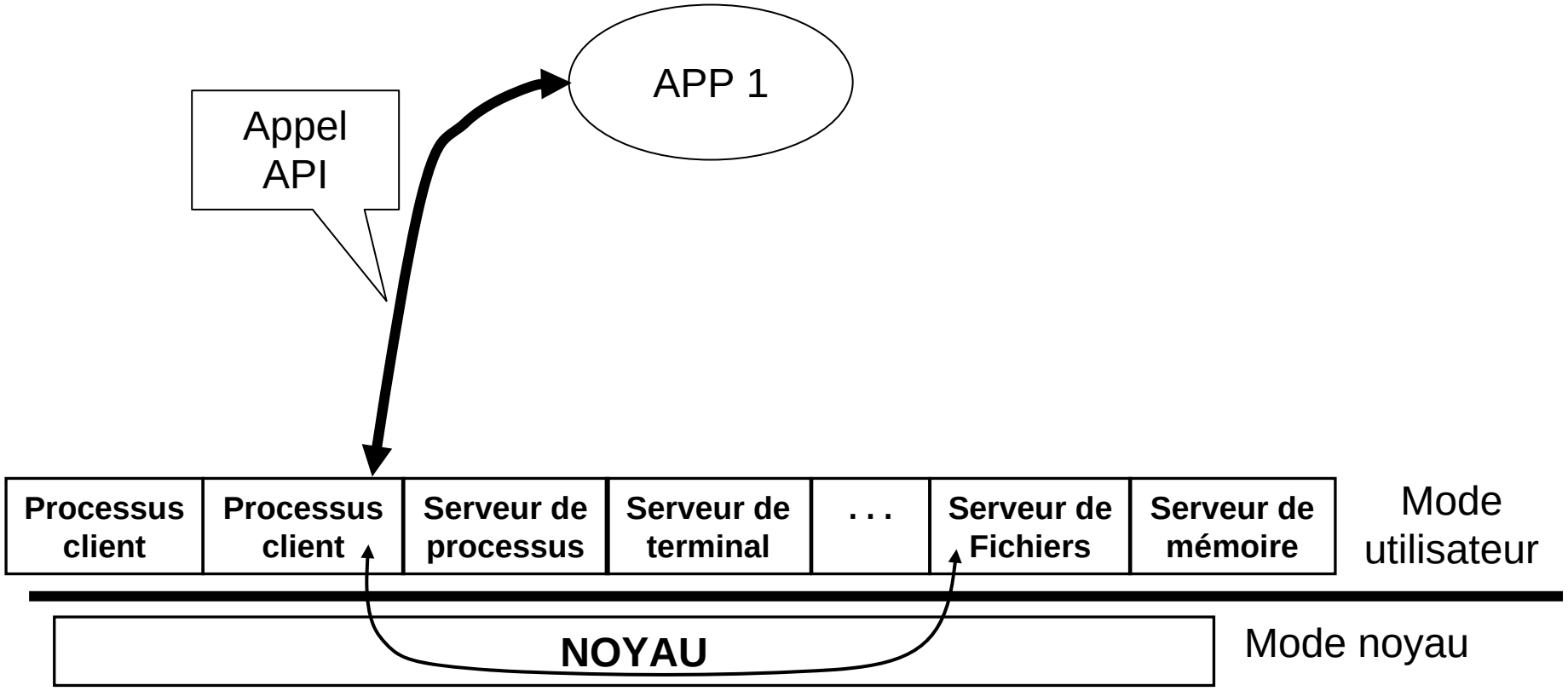
- Les systèmes monolithiques :
  - Structure unique
  - Collection de procédures
    - Chacune peut appeler les autres
    - Chaque procédure est visible de toutes les autres
  - Inconvénient :
    - Taille du noyau
  - Avantage :
    - Performance
  - Alternative
    - Systèmes monolithiques modulaires





Architectures

Les mécanisme client – serveur :



Le système d'exploitation

- Un système d'exploitation est un ensemble de programmes dont le but est de faciliter l'utilisation du matériel.
- Il assure la gestion des ressources par l'utilisation d'objets gérés par le système.
- Les systèmes peuvent être structurés de différentes façons

Entités d'exécution

Pour l'utilisateur une tâche c'est :

- Une application
- Un ensemble d'applications

Entités d'exécution

- Une application c'est :
  - Un ou plusieurs fichier(s) exécutable(s) chargé(s) en mémoire afin d'être exécuté(s).
  - Un ensemble de programmes exécutés : Les processus

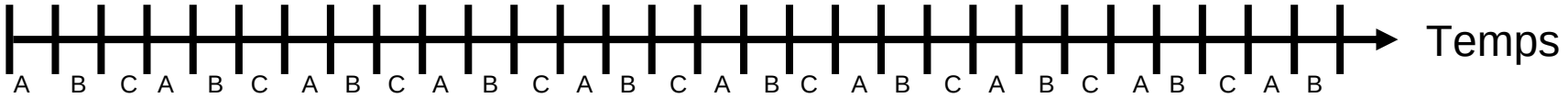
- La notion de processus est fondamentale dans tous les systèmes d'exploitation modernes.

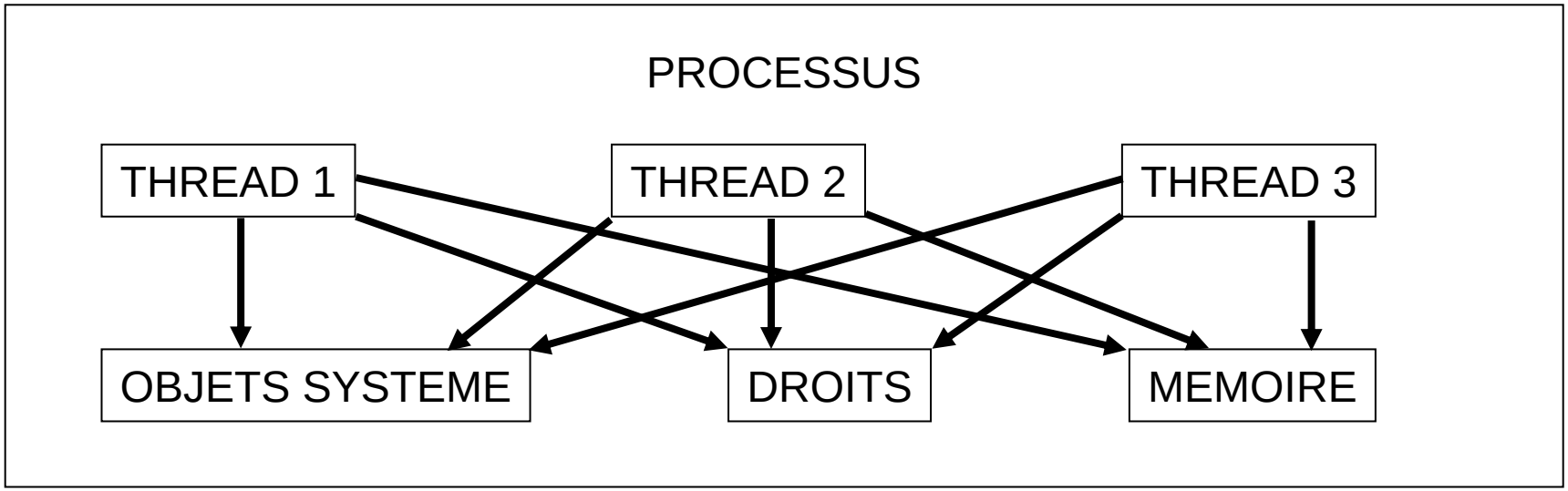
Entités d'exécution

- Les processus
  - Identité.
  - Code exécutable.
  - Une zone de mémoire propre.
  - Des données.
  - Des objets système manipulés par les appels système effectués par le code du processus.
  - Des droits d'accès aux objets du système qu'il manipule.
  - Un compteur ordinal.
  - Un contexte d'exécution.
    - Sous Unix : fork
    - Sous Windows : CreateProcess

Entités d'exécution

- Lors des commutations de processus, le temps est important





- Pour réduire ce temps de commutation : les threads
- La commutation des threads prend moins de temps que celle des processus.

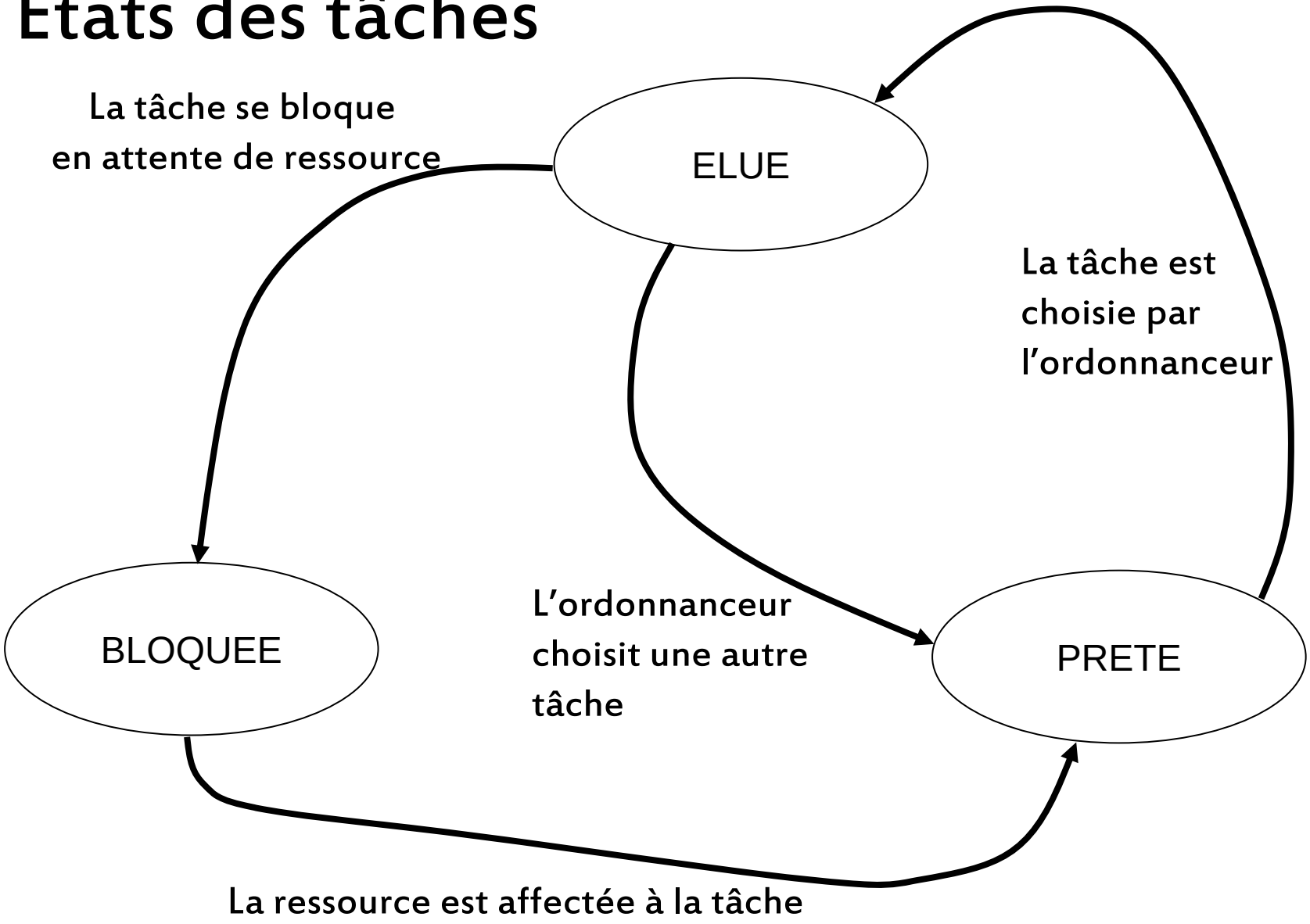
Entités d'exécution

- Propriétés des systèmes multi – processus multi – threads
  - Un processus est composé d'au moins un thread.
  - Un thread appartient à un et un seul processus.
  - Si un thread meurt, les autres threads composant le processus auquel appartient le thread mourant continuent d'exister. Si le processus meurt, tous les threads composant ce processus meurt en même temps que lui.
  - Un thread possède une identité attribué par le système.
  - Tous les threads du processus fonctionnent dans l'espace d'adressage du processus.

Entités d'exécution

- Propriétés des systèmes multi – processus multi – threads
  - L'exécution d'un thread s'effectue en concurrence avec les autres threads qu'ils appartiennent au même processus ou pas.
  - Les variables globales et les descripteurs d'objets d'un processus sont partagés par tous les threads du processus.
  - Un thread peut disposer d'une zone de mémoire qui lui est propre, explicitement sur sa demande et implicitement par les variables locales des fonctions qu'il exécute.

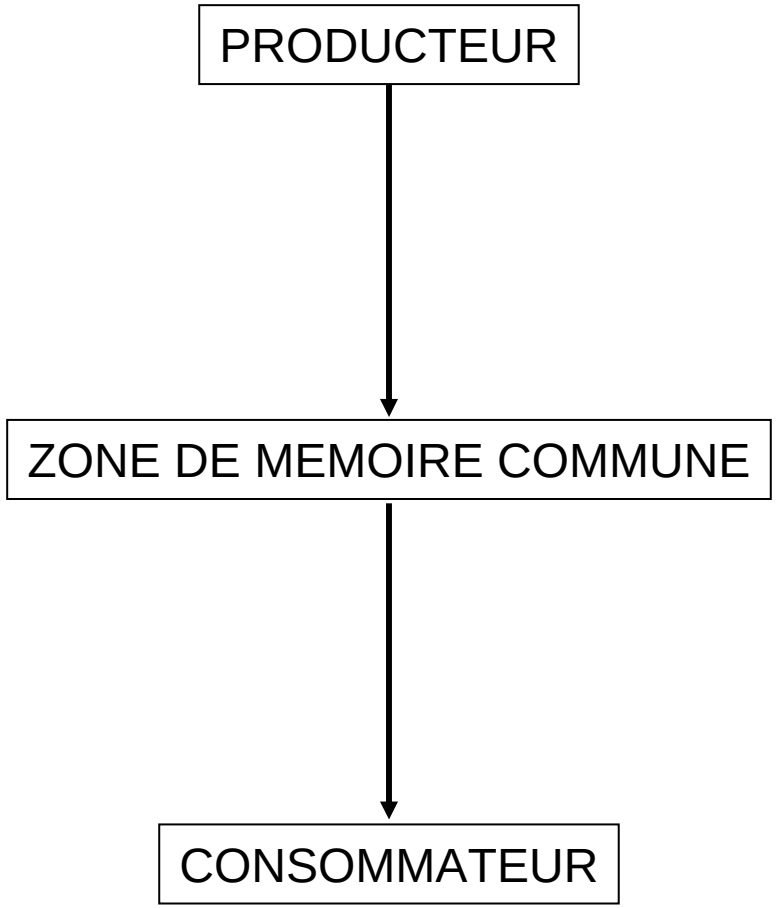
Etats des tâches



Communication entre tâches

- Les tâches communiquent entre elles
  - Par des mécanismes du système d'exploitation pour les processus :
    - les tubes
    - les boites aux lettres
    - la mémoire partagée
  - Par accès à des zones mémoires communes au sein du processus pour les threads :
    - Nécessité des mécanismes d'exclusions mutuelles

Contraintes



- Accès concurrents à la mémoire commune

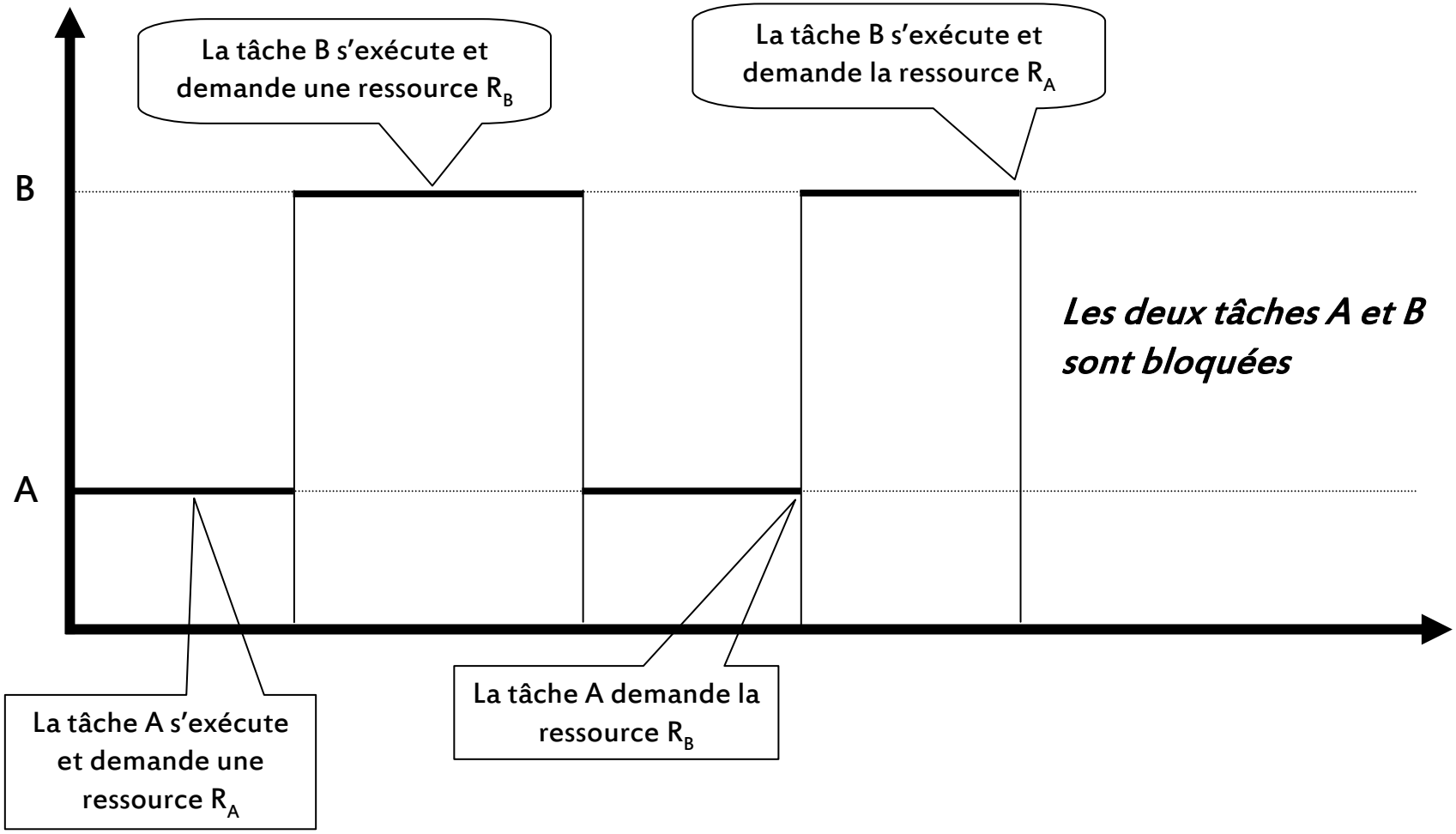
Mécanisme de synchronisation

- Mutex (MUTual EXclusion): busy, wait, over
- Sémaphores (mutex avec compteur)
- Sections critiques
- Les rendez-vous (ADA)
- Les objets partagés

- Les boites aux lettres
- Les tubes

Interblocage

Deux tâches peuvent se bloquer mutuellement :

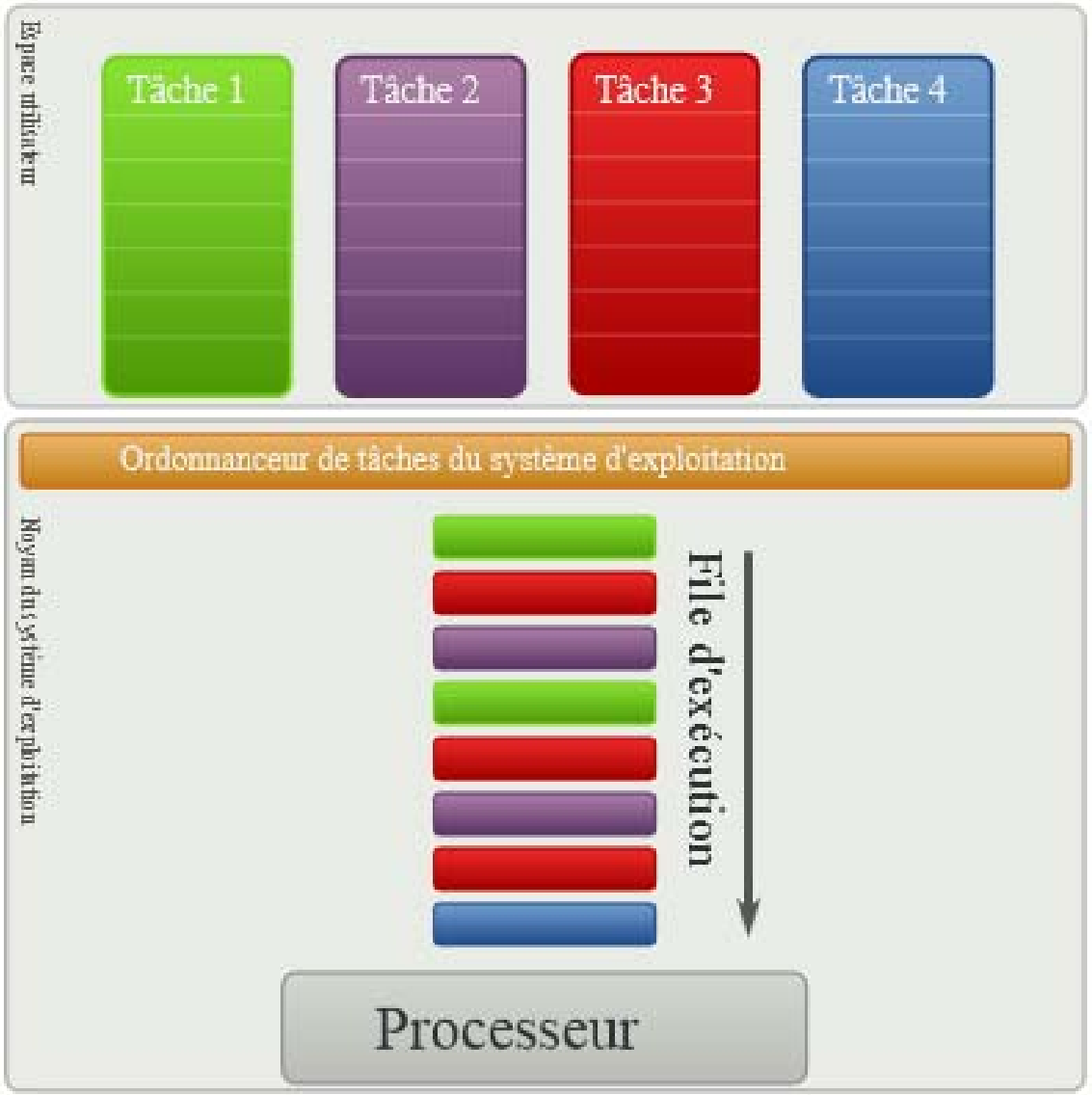


Ordonnancement

- L'ordonnanceur doit :
  - S'assurer que chaque tâche reçoit sa part du temps processeur.
  - Optimiser l'utilisation du processeur
  - Minimiser les temps réponses pour les utilisateurs en mode interactif.
  - Maximiser le nombre de travaux effectués

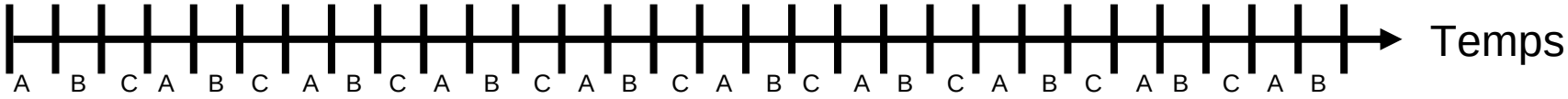
- Il existe plusieurs politiques d'ordonnancement
  - FIFO
  - Tourniquet
  - Priorité
  - Préemption
  - Coopératif

Ordonnancement

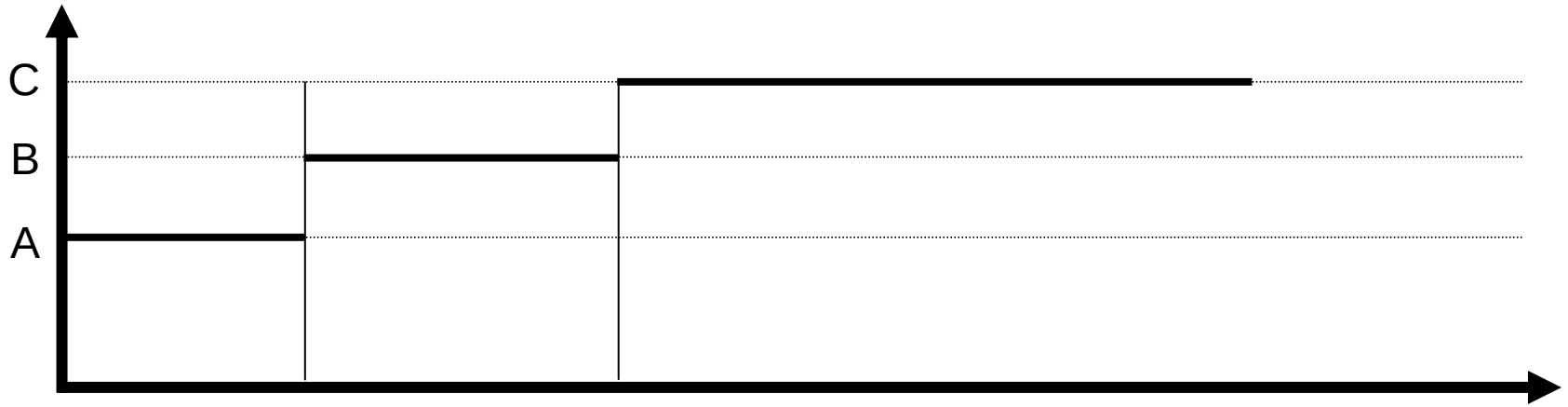


Cas d'école

- Ordonnancement en temps partagé
  - Soit trois processus A, B et C. Ils s'exécutent les uns après les autres pendant le même temps. La ressource processeur est partagée entre ces trois processus

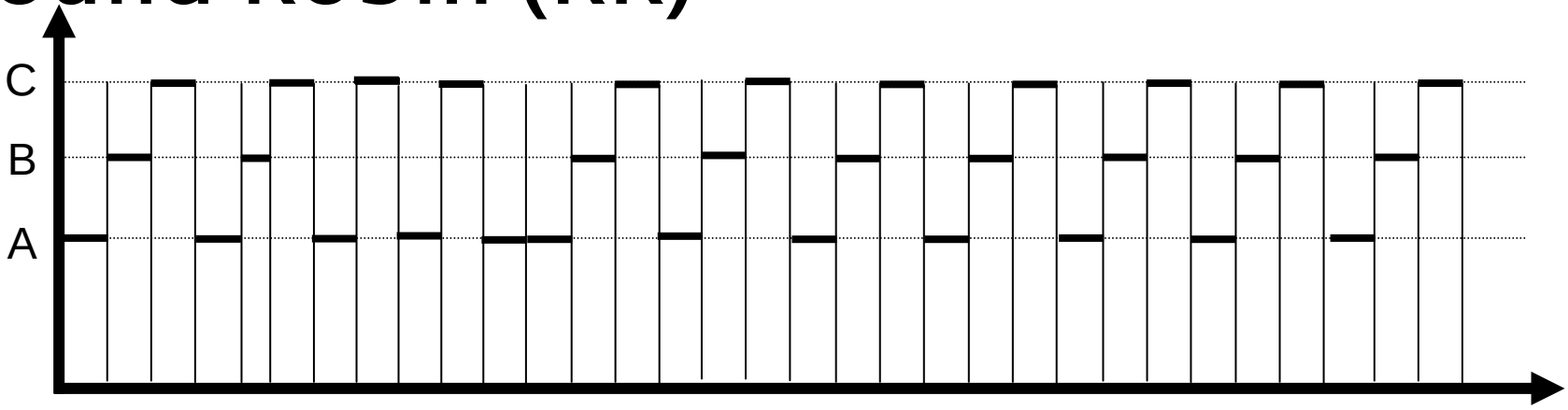


Ordonnancement FIFO



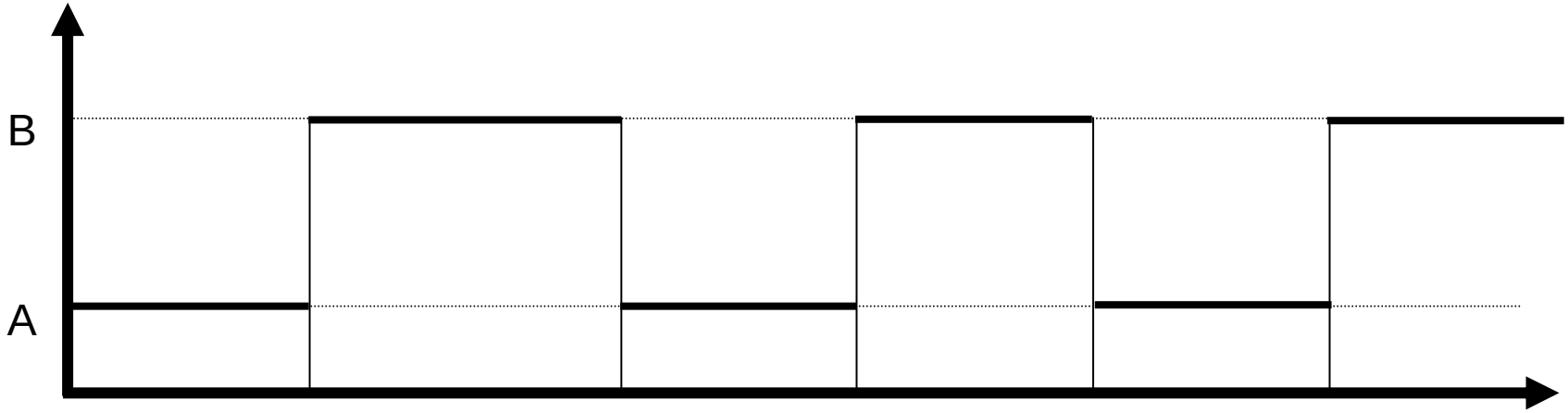
- Les tâches sont exécutées les unes après les autres.
- Une tâche ayant la ressource processeur ne la rend que :
  - volontairement,
  - par blocage de la tâche suite à une demande d'entrées – sorties,
  - par terminaison de la tâche.

Round Robin (RR)



- Temps partagé: la fenêtre d'exécution est appelée **quantum de temps** ou *time slice*.
  - Le système utilise une liste et ordonnance les tâches suivants leurs positions dans cette liste.
  - A la fin de son time slice, la tâche est remise en fin de liste.
- Le quantum de temps doit être un compromis entre les temps utiles de travail (temps d'exécution des tâches) et le temps pris pour commuter les tâches.

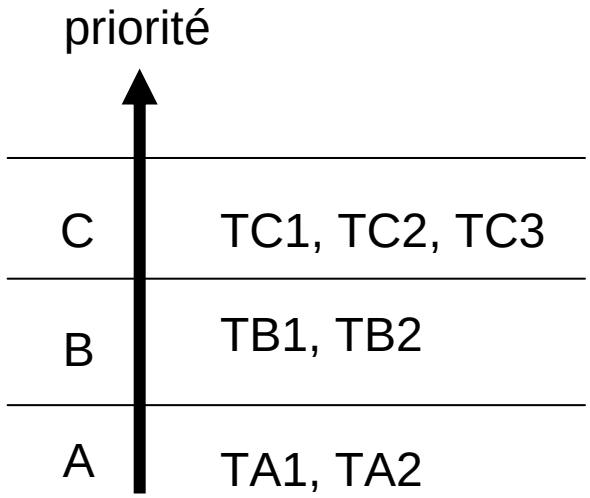
Ordonnancement par priorité



- Chaque tâche dispose d'une valeur reflétant sa priorité. Le système choisit la tâche de priorité la plus haute (sous réserve qu'elle dispose de toutes les ressources).
- Le système dispose de files par niveau de priorité. Dans une file de niveau de priorité P sont mises les tâches de même priorité P.

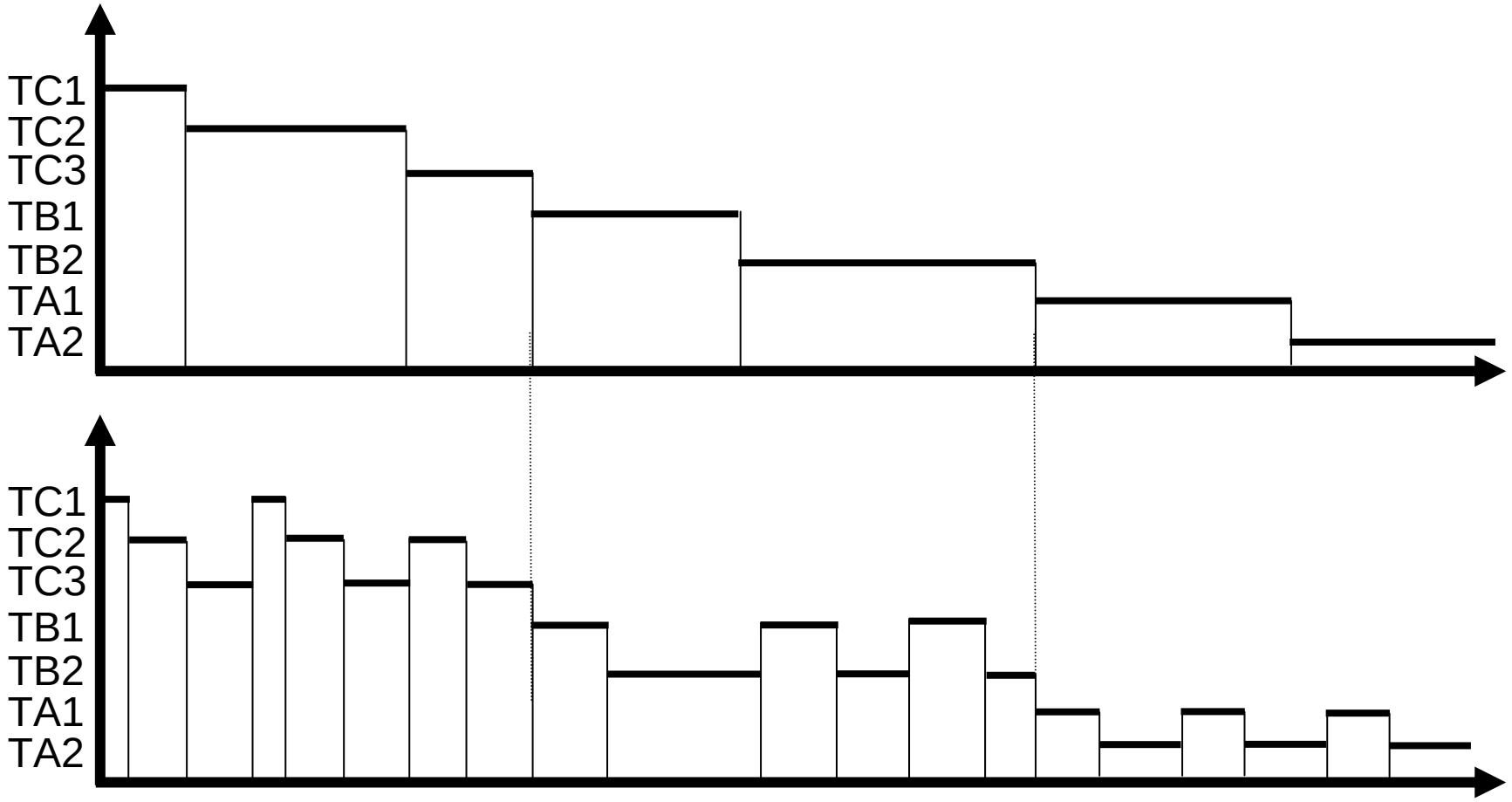
Ordonnancement par priorité

- Si plusieurs tâches ont le même niveau de priorité et que ce niveau est le plus haut, une seconde politique d'ordonnancement doit exister (FIFO ou RR)



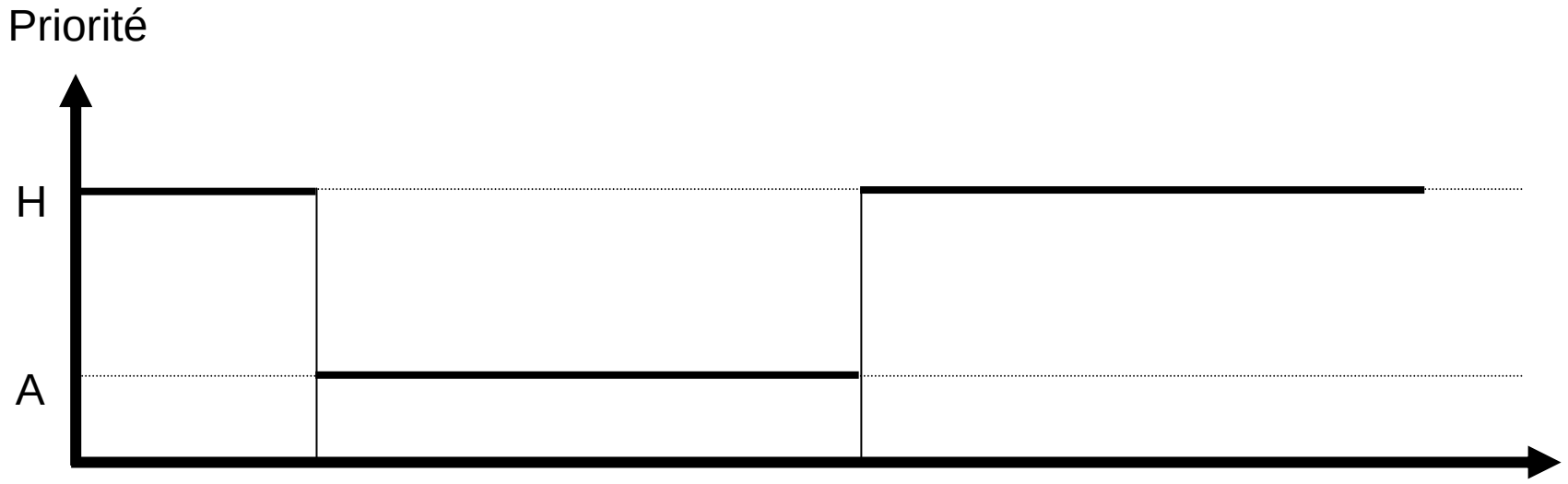
- Quel est l'ordonnancement des tâches ci-dessus
  - en tourniquet
  - en fifo

Ordonnancement par priorité



Ordonnancement par préemption

- Si une tâche de haute priorité H demande une ressource au système et se retrouve en l'état bloquée, la tâche A de priorité immédiatement inférieur à H s'exécute. Lors de la libération de la ressource par A, la tâche de haute priorité H s'exécute immédiatement préemptant toutes les tâches de priorité inférieure.



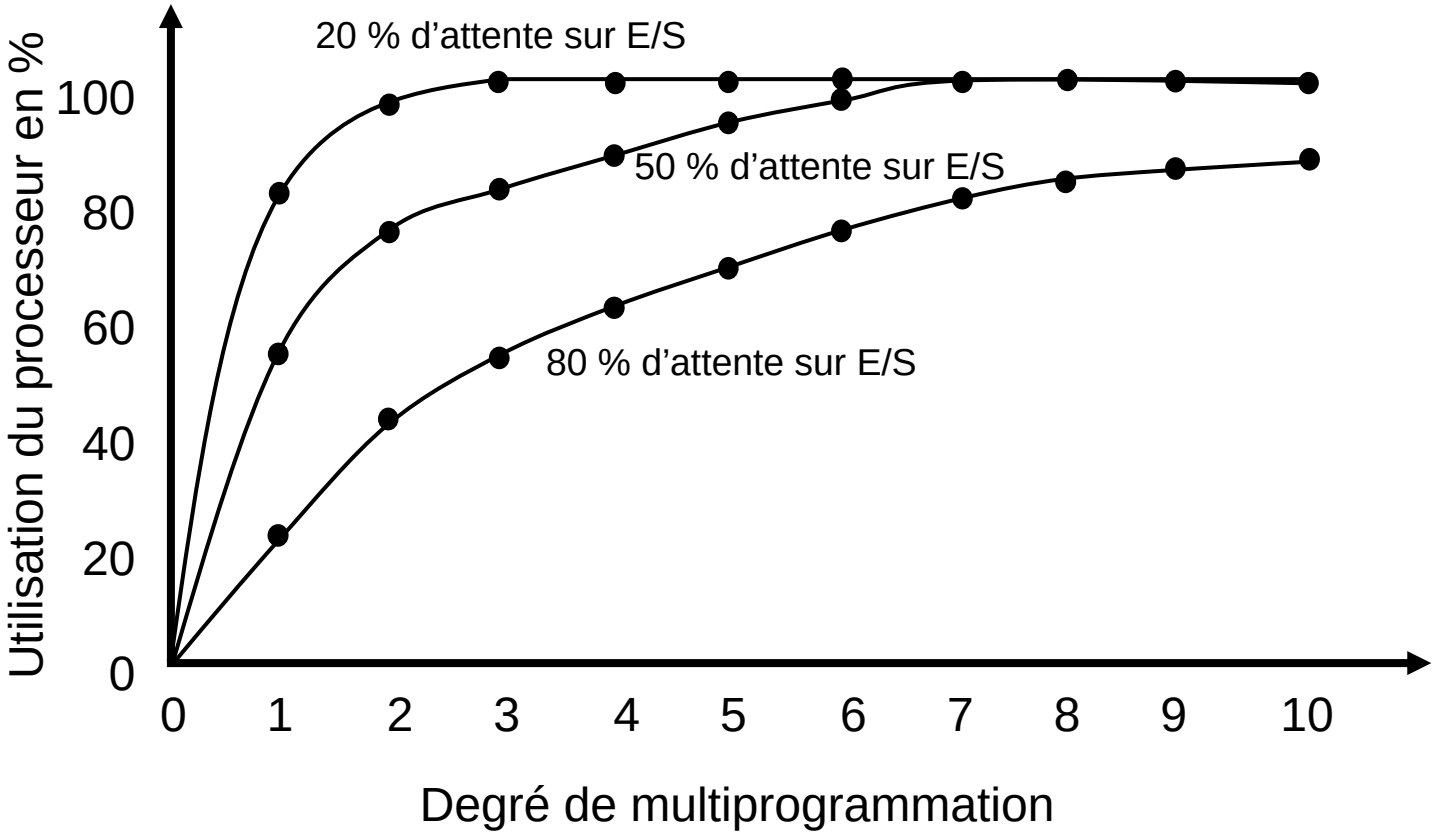
Ordonnancement coopératif

- Dans ce type de système, ce sont les tâches qui décident de libérer volontairement le processeur (sauf si elles sont bloquées par une E/S).

- Exemple : Windows 3.X

- Ce mode est dangereux car si une application est dans une boucle sans fin, elle ne rendra pas le processeur et empêchera les autres tâches de s'exécuter.

Utilisation du temps UC



L'ordonnancement

- Processus
- Threads
  - Problèmes d'accès aux ressources
  - Problèmes d'inter – blocage

- L'ordonnancement
  - Exécution des tâches

- L'ensemble de ces problèmes influence la conception des logiciels.

La mémoire cache

- Le temps passé à attendre une réponse de la mémoire peut pénaliser fortement le temps d'exécution d'un programme.

- Les références aux données et aux instructions ne sont pas indépendantes.
  - Les programmes ont tendance à réutiliser les données et les instructions qu'ils ont utilisées récemment.

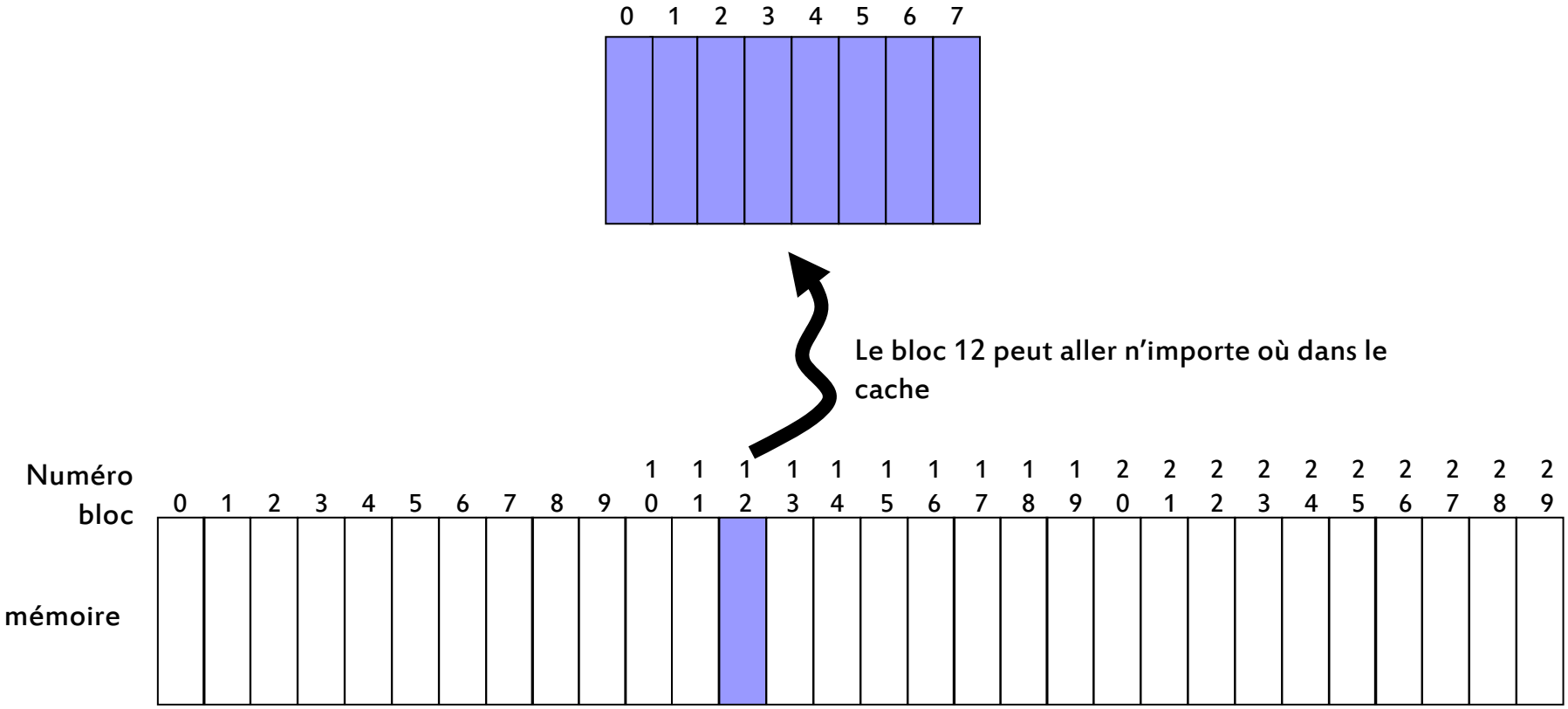
- Localité spatiale
  - les éléments dont les adresses sont proches les unes des autres auront tendance à être référencés dans un temps rapproché
- Localité temporelle
  - les éléments auxquels on a eu accès récemment seront probablement accédés dans un futur proche (ex: boucles).

La mémoire cache

- Où peut-on placer un bloc dans le cache ?
  - Placement de bloc
- Comment trouver un bloc s'il est présent dans le cache ?
  - Identification de bloc
- Quel bloc doit être remplacé en cas d'échec ?
  - Remplacement de bloc
- Qu'arrive-t-il lors d'une écriture?
  - Stratégie d'écriture

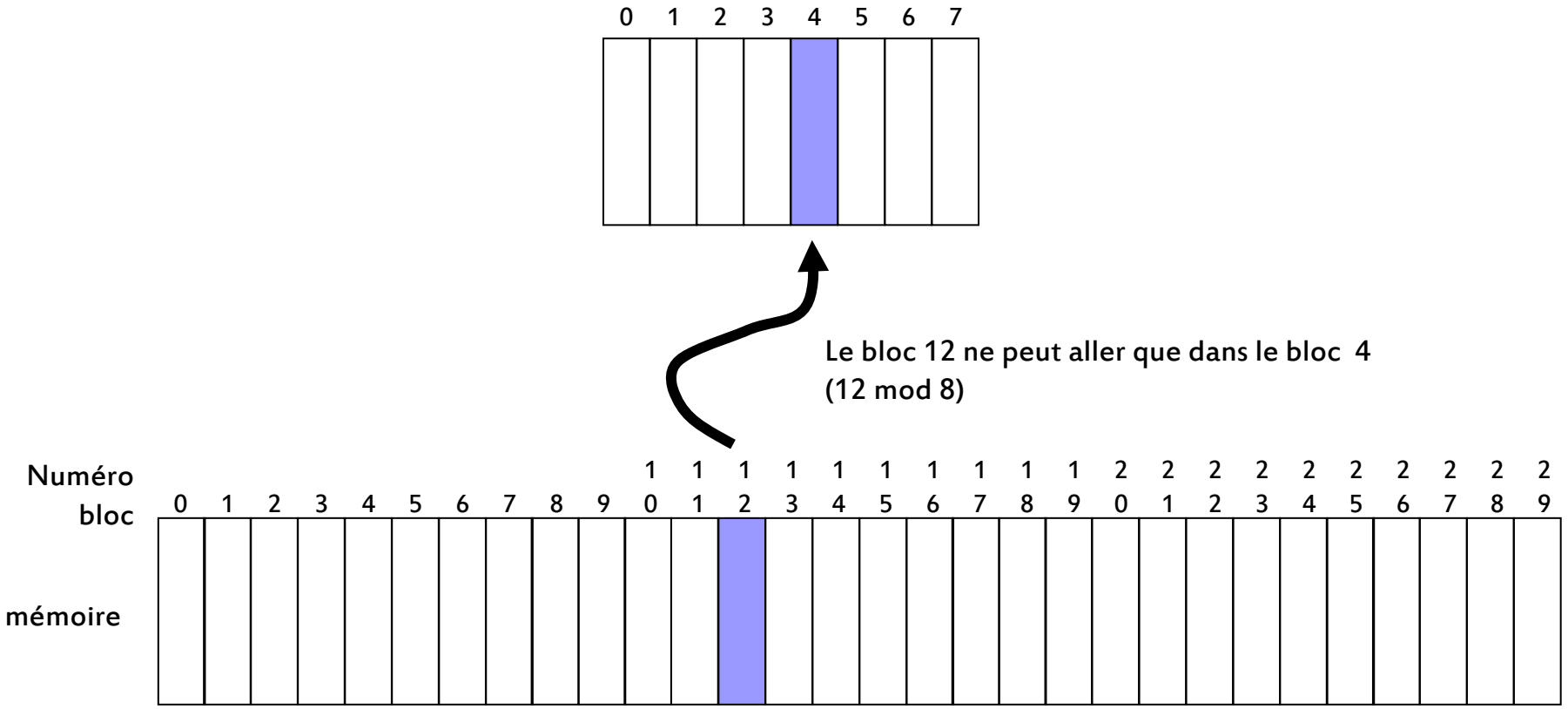
La mémoire cache

- Cache totalement associatif



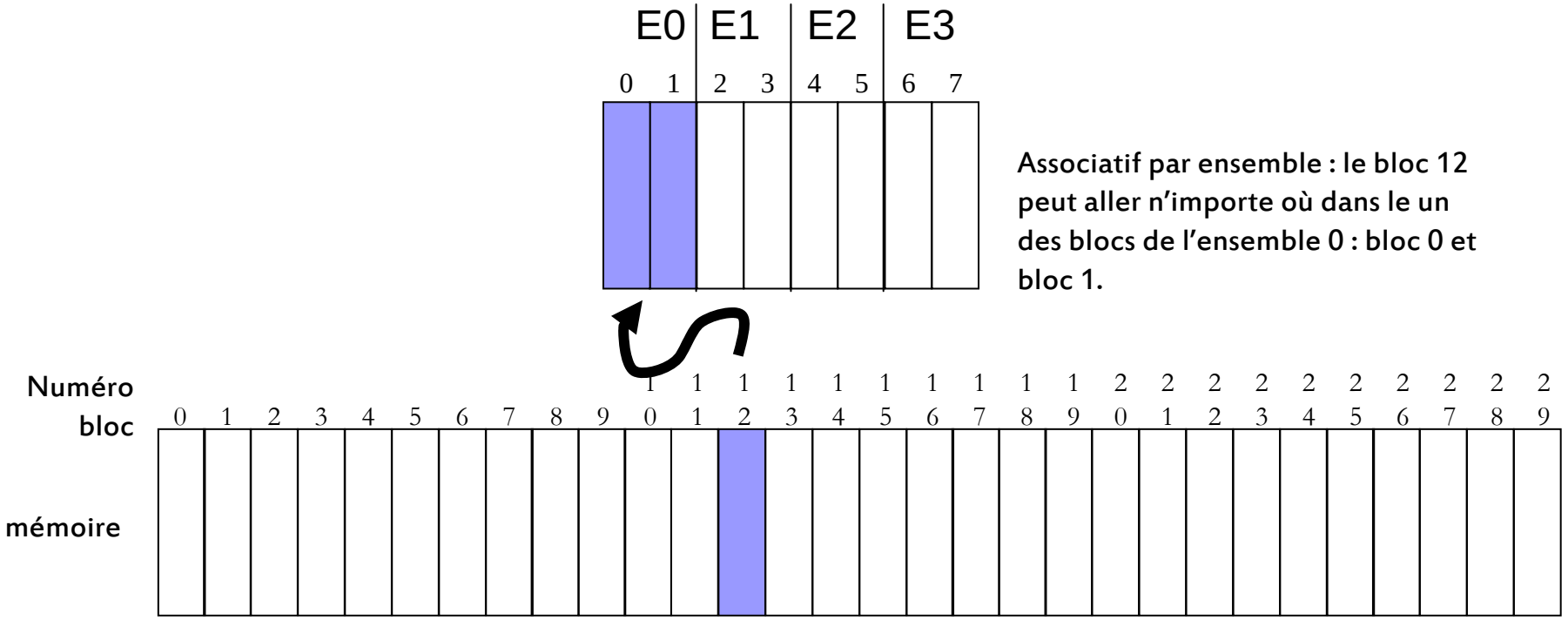
La mémoire cache

- Cache à correspondance directe



La mémoire cache

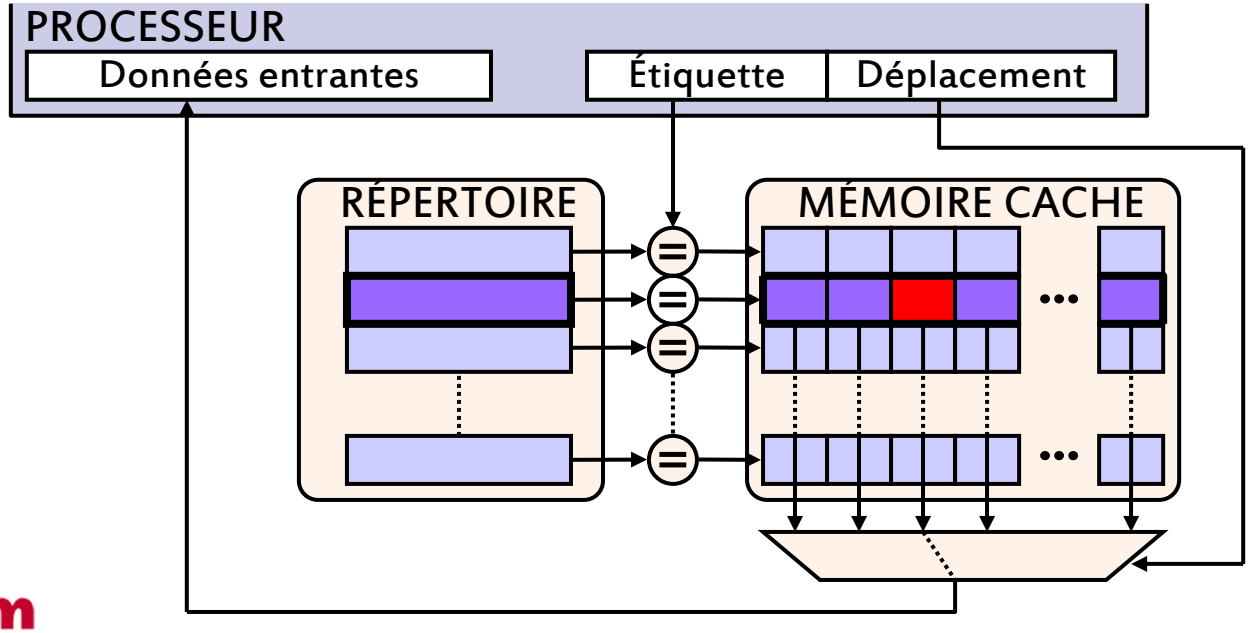
- Cache associatif par ensemble de blocs
  - Le bloc mémoire peut être placé dans un ensemble de blocs :
    - (numéro bloc) modulo (nombre d'ensemble dans le cache)



Associatif par ensemble : le bloc 12 peut aller n'importe où dans le un des blocs de l'ensemble 0 : bloc 0 et bloc 1.

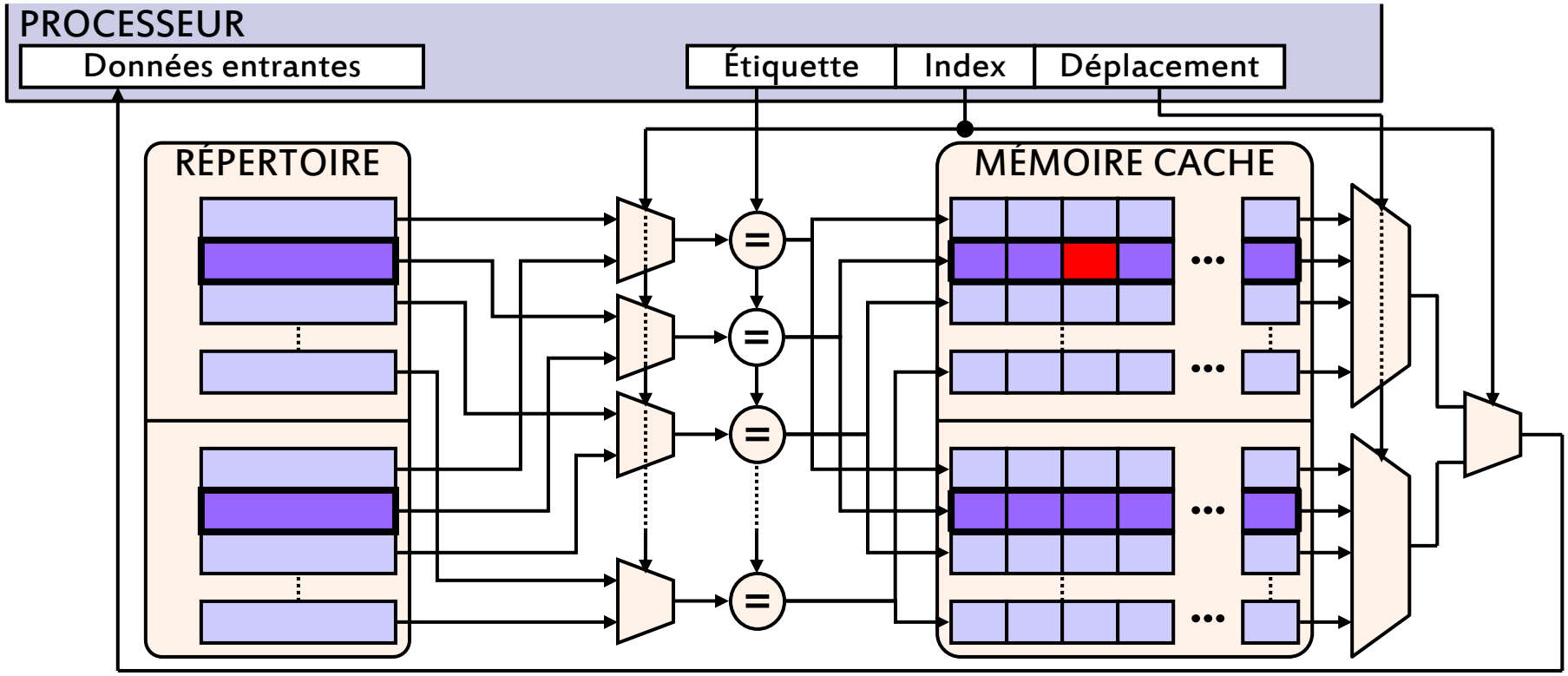
La mémoire cache

- Un répertoire est associé à chaque cache associatif. Le répertoire contient le numéro (adresse) de chaque bloc dans le cache, appelé étiquette. Toute les étiquettes dans le répertoire sont examinées en parallèle.
- Pour un cache totalement associatif, chaque adresse contient donc une étiquette et un déplacement dans le bloc.



La mémoire cache

- Pour un cache associatif par ensemble, une adresse cherchée par le processeur peut être partagée en 3 champs:
  - le déplacement est l'adresse dans le bloc du mot cherché
  - l'index identifie l'ensemble
  - l'étiquette est la partie de l'adresse du bloc utilisée pour la comparaison.



La mémoire cache

- Quand un échec intervient, le contrôleur de cache doit choisir un bloc à remplacer par la donnée recherchée.
  - Pour les caches à correspondance directe, ce choix est obligatoire.
  - Pour les caches à associativité totale ou par ensemble, plusieurs algorithmes existent:
    - Le hasard.
    - Moins fréquemment utilisée (LFU pour least frequently used).
    - Le plus ancien (LRU pour least recently used)
    - Pas le dernier (NLU pour Not Last Used)

LRU

Algorithme de remplacement d'une page non récemment utilisée

Périodiquement, le système remet à zéro les bits de référencement de la page. Lors des défauts de page, le système répartie toutes les pages en fonction des différentes valeurs du bit de lecture et des bits de modification :

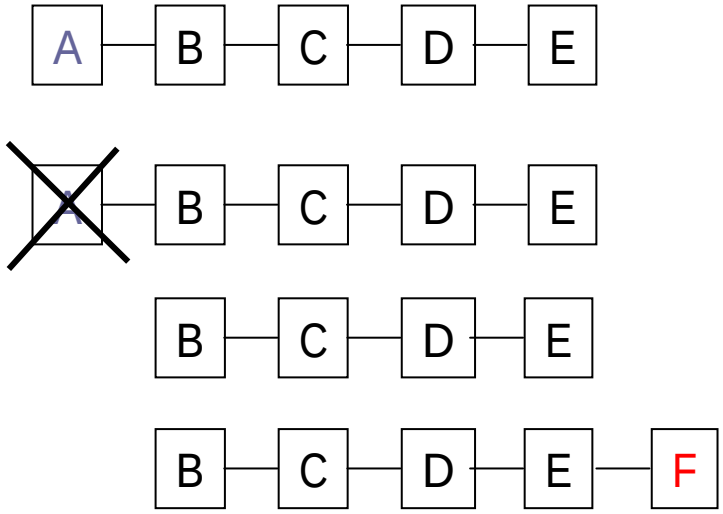
- Catégorie 0 : non référencée, non modifiée
- Catégorie 1 : non référencée, modifiée
- Catégorie 2 : référencée, non modifiée
- Catégorie 3 : référencée, modifiée

L'algorithme retire une page au hasard de la catégorie non vide qui a le plus petit numéro

FIFO

Chaque page est répertoriée dans une liste dont l'ordre donne la chronologie des pages :

- La première de la liste est la plus ancienne
- La dernière de la liste est la plus récente



Variante : si le bit de référence de A est à 0, A est supprimé de la liste, s'il est à 1, il est remis à 0 et A est mis en fin de liste. Puis la recherche continue. Si aucune page n'a son bit de référence à 0, A sera alors supprimé. C'est l'algorithme de seconde chance.

La mémoire cache

- Stratégies de mise à jour
  - Write through (Ecriture simultanée)
    - Mise à jour simultanée du cache et de la mémoire principale
  - Write back (Ecriture différée)
    - Consiste à écrire uniquement dans le bloc du cache. Le bloc modifié du cache est recopié en mémoire principale uniquement quand il est remplacé.
    - Un bit modifié dans le répertoire du cache indique si le bloc a été modifié.

La mémoire cache

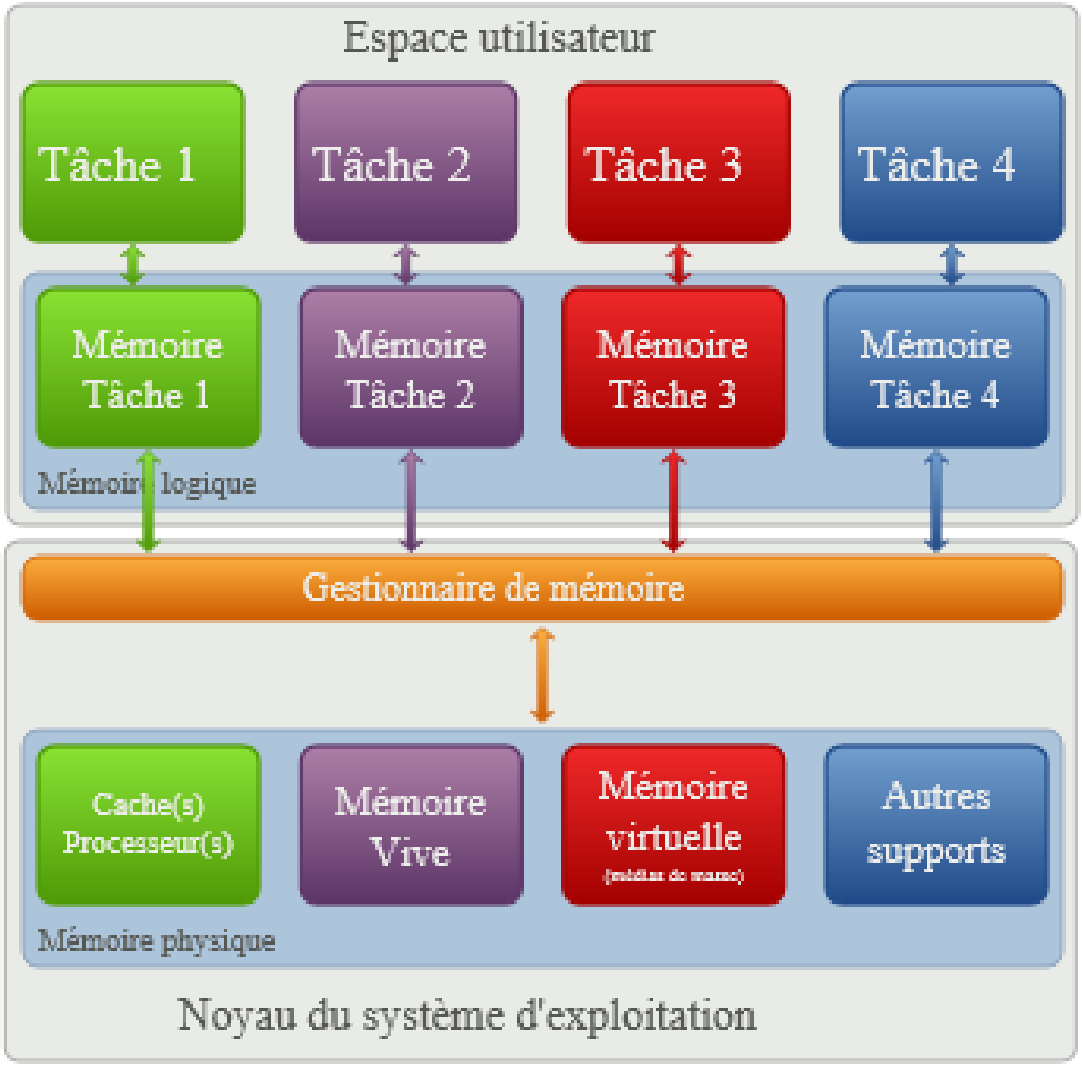
- Le processeur cherche un mot d'abord dans le cache. En cas d'échec (cache miss), le mot est cherché en mémoire et son bloc est copié dans le cache pour un accès futur. En cas de succès (cache hit), la mémoire principale n'est pas accédée.
- Différents types d'échecs existent:
  - Échec obligatoire : lors de son premier accès, un bloc n'est pas dans le cache.
  - Échec de capacité: si le cache ne peut contenir tous les blocs nécessaires pendant l'exécution d'un programme, des blocs seront écartés.
  - Échec de conflit : dans des caches associatifs par ensemble ou à correspondance directe, un bloc peut être écarté et rappelé plus tard si trop de blocs doivent être placés dans son ensemble.

La mémoire virtuelle

- Problématique
  - Système multi – utilisateur
  - Système multi – tâche.
  - De grandes capacités mémoire dépassant la mémoire physique de la machine.

- Solution
  - la mémoire virtuelle
  - Faire croire au logiciel qu'il possède plus de mémoire que n'en possède réellement la machine.

La mémoire virtuelle



La mémoire virtuelle

Adressage logique

Spécifie l'adresse d'un opérande et/ou d'une instruction

Chaque adresse logique est constituée d'un segment et d'un offset

Adressage linéaire

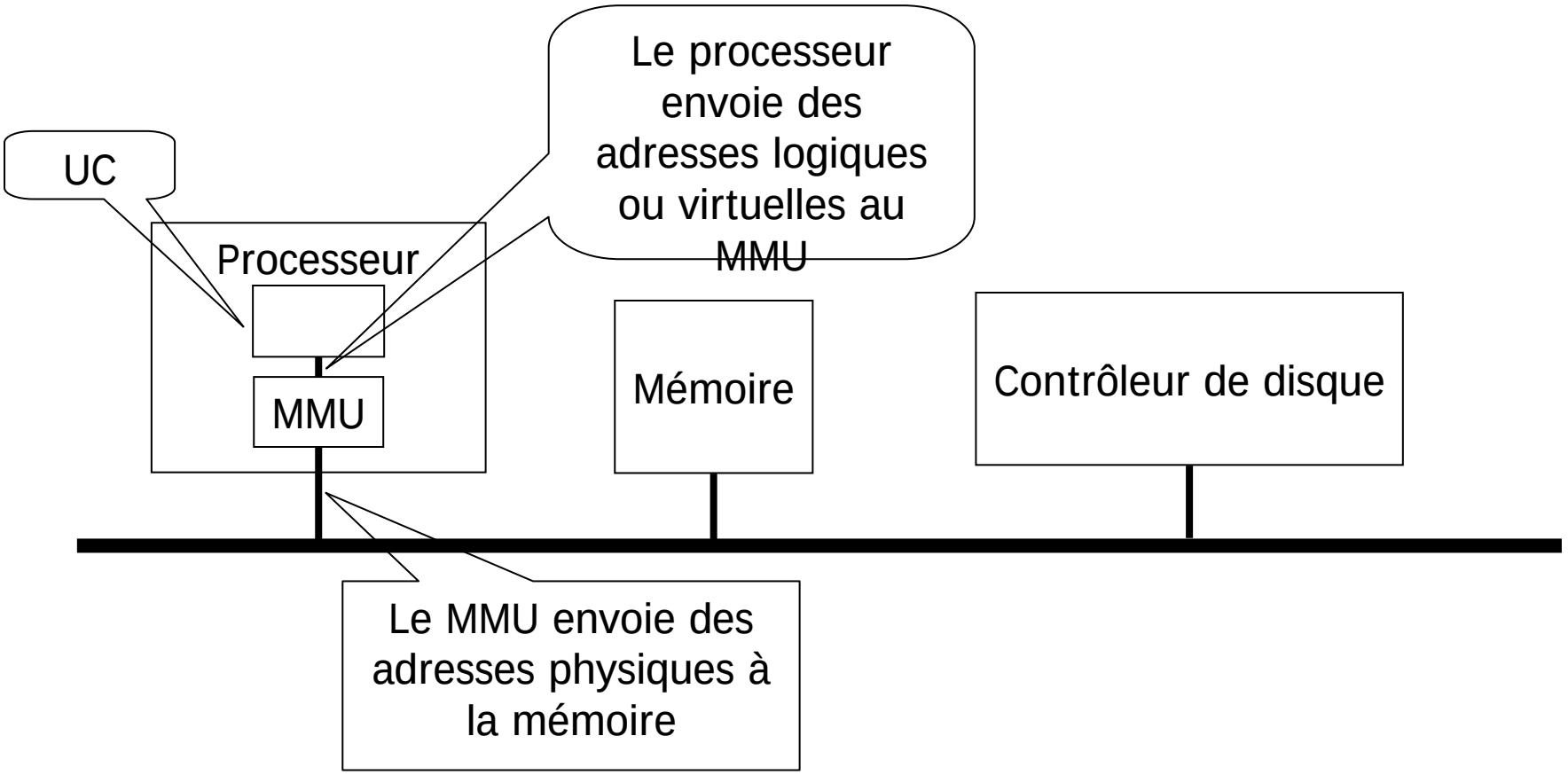
Un entier 32bits non signé permettant d'adresser n'importe quelle zone mémoire (CPU x86)

Adressage physique

Adresse la cellule mémoire

Correspond aux signaux électriques

La mémoire virtuelle



La mémoire virtuelle

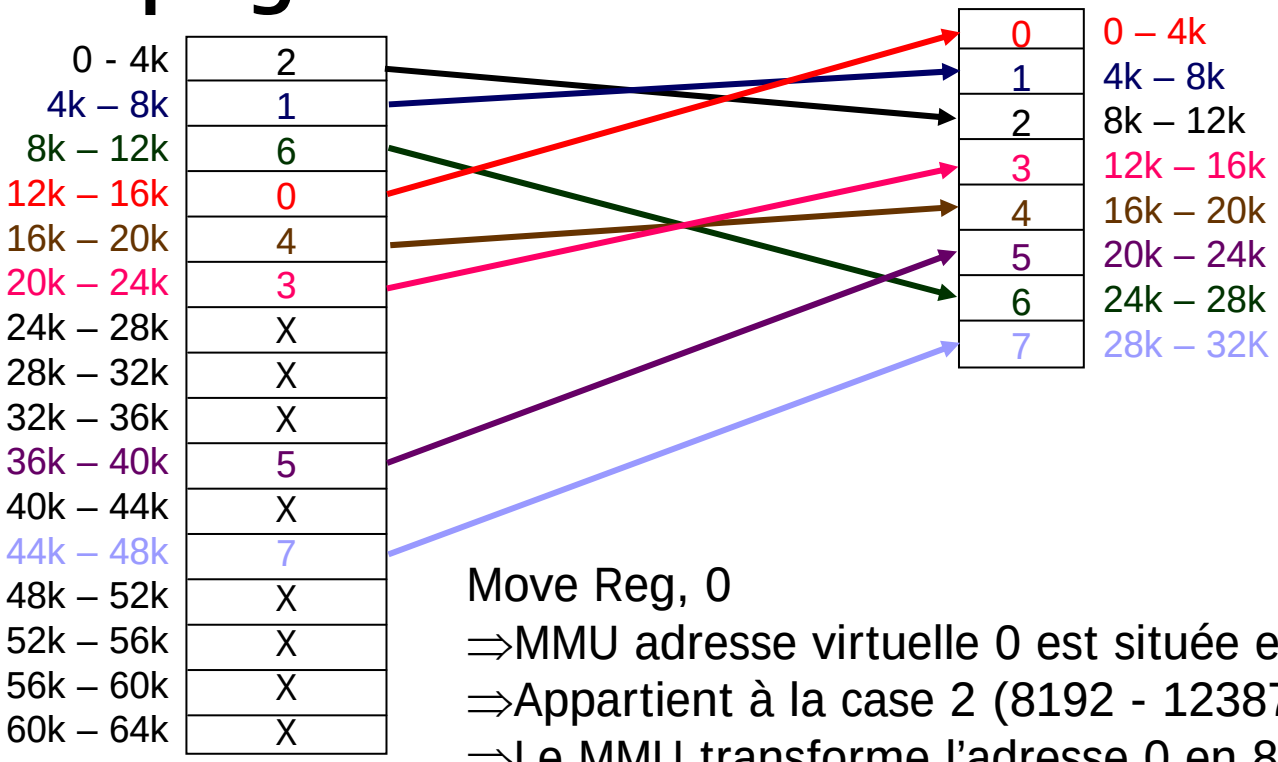
Rôles de la MMU

- La traduction d'adresses logiques en adresses linéaires par l'unité de segmentation
- La traduction d'adresse linéaire en adresses physiques par l'unité de pagination
- Le contrôle de tampon
- L'arbitrage du bus
- La protection de la mémoire (généralement cette fonction est faite par la MPU (Memory Protection Unit))

La pagination

- L'espace d'adressage virtuel :
  - Les adresses manipulés par les processus.
  - L'ensemble des adresses virtuelles.
- La mémoire virtuelle utilise la technique de la pagination
  - Cette technique consiste a diviser l'espace d'adressage virtuel en petites unités appelées pages. Elles sont toutes de la même taille (en général 4Ko). Les pages correspondantes à la mémoire réelle s'appellent des cases mémoire (pages frames).
- Les pages non utilisées sont mises sur le disque dur et remplacées par d'autres (lors des défauts de page) dont le contenu est utile aux programmes en cours : le « va et vient » (*swap*).

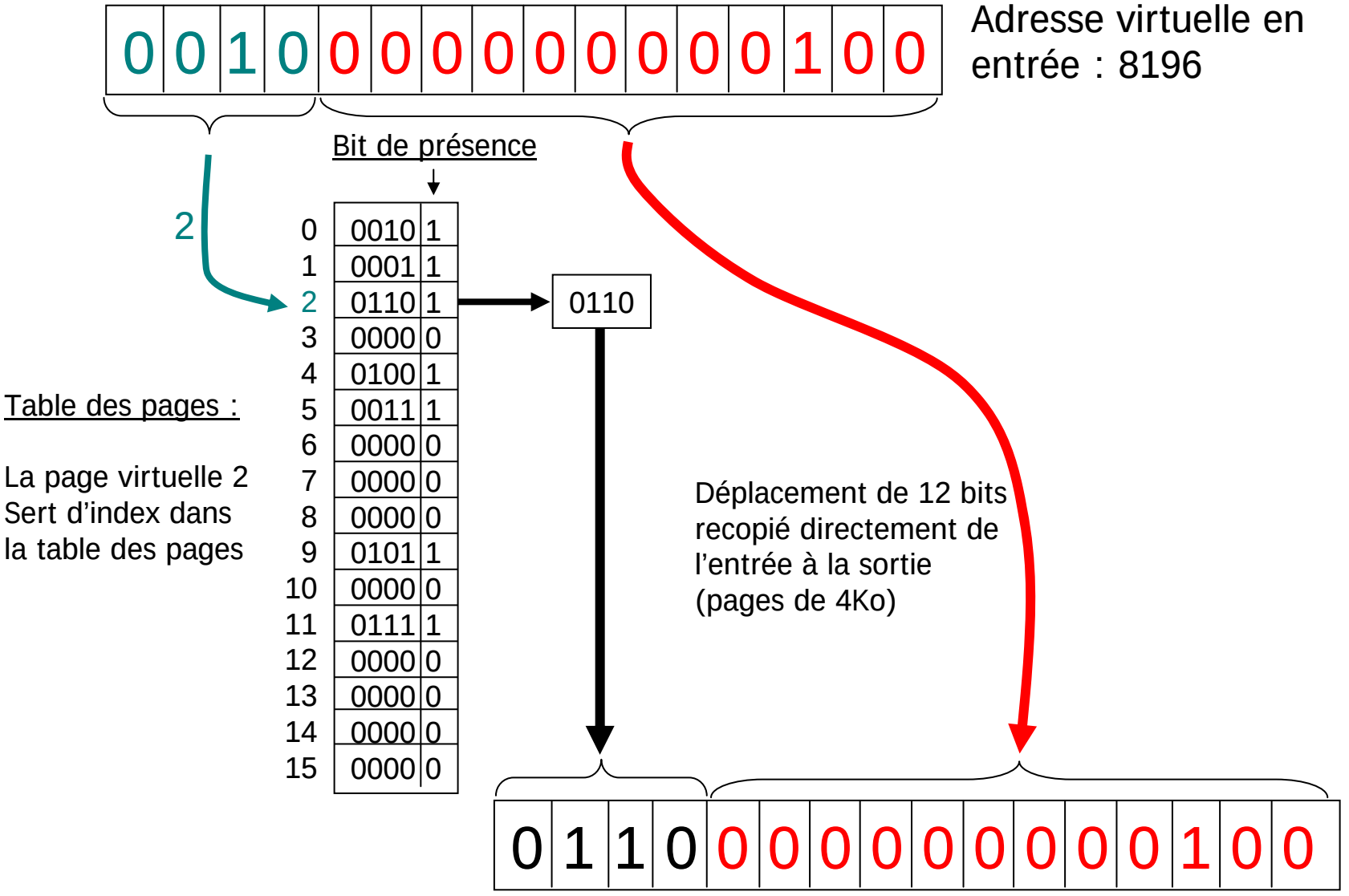
La pagination



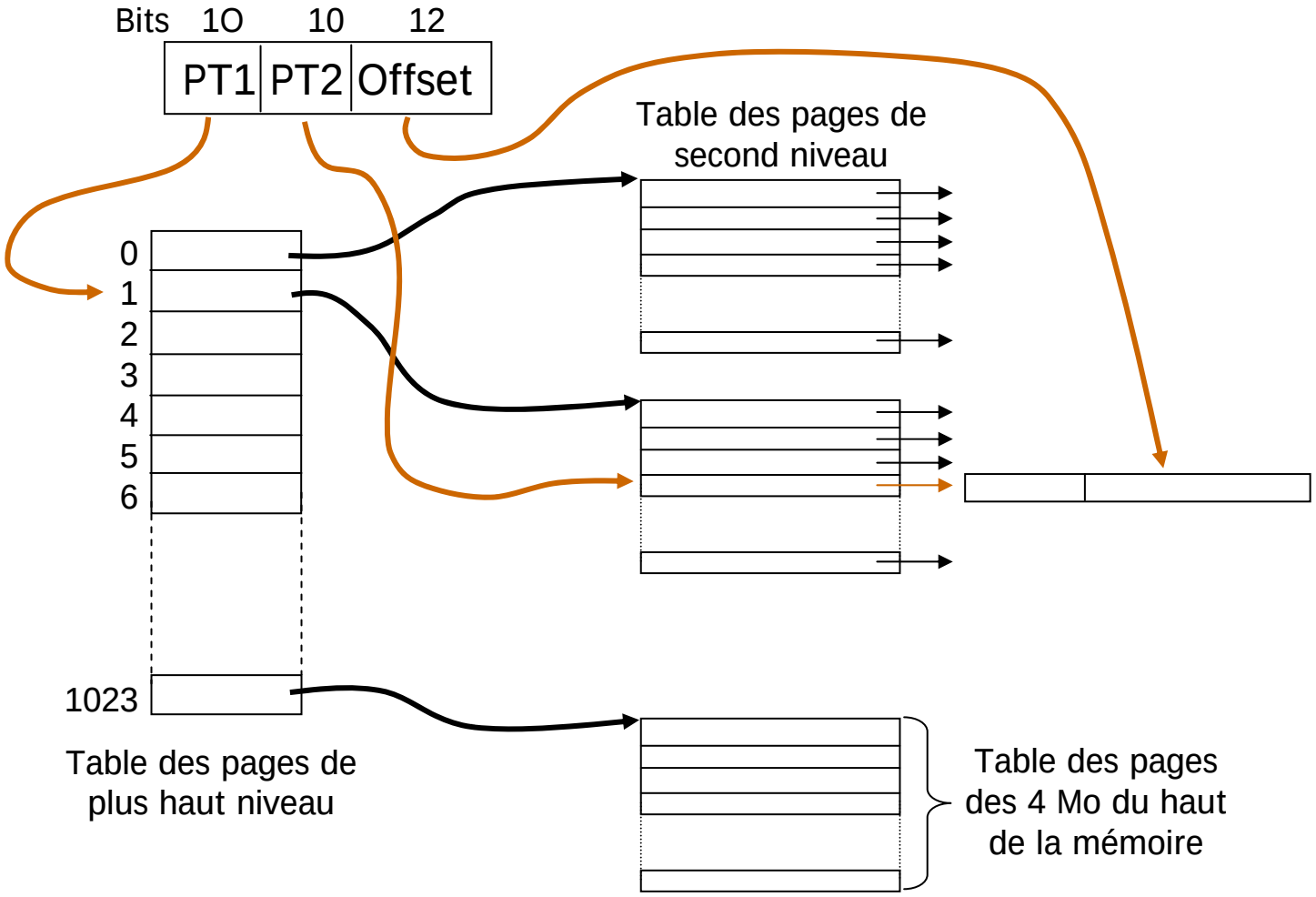
Move Reg, 0
⇒MMU adresse virtuelle 0 est située en page 0 (0 - 4095)
⇒Appartient à la case 2 (8192 - 12387)
⇒Le MMU transforme l'adresse 0 en 8192

Move Reg, 8192 => Move Reg, 24576
⇒8192 est situé dans la page virtuelle 2 qui est mappée à la case 24576 - 28671

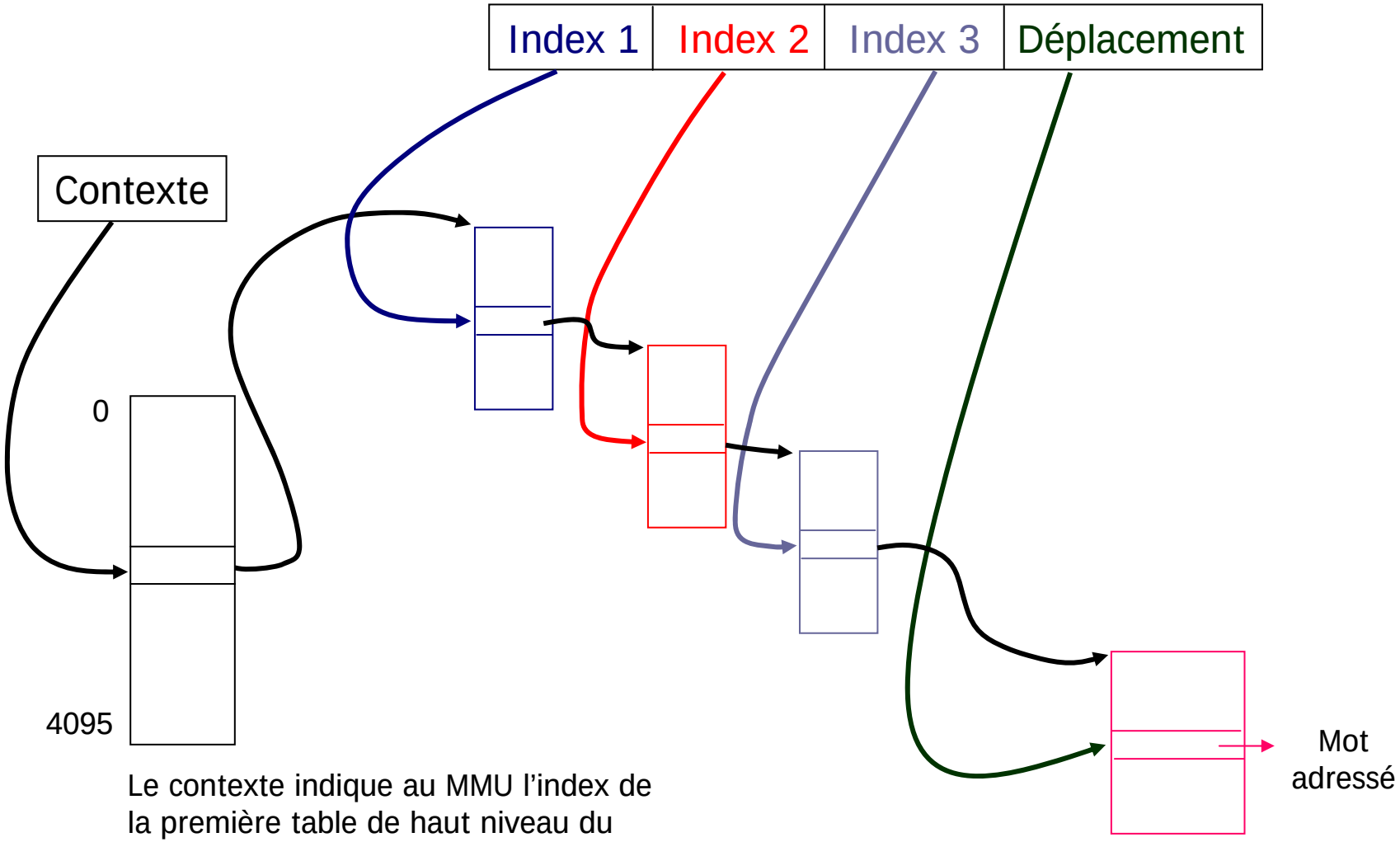
La pagination



La mémoire virtuelle à plusieurs niveaux



Le SPARC

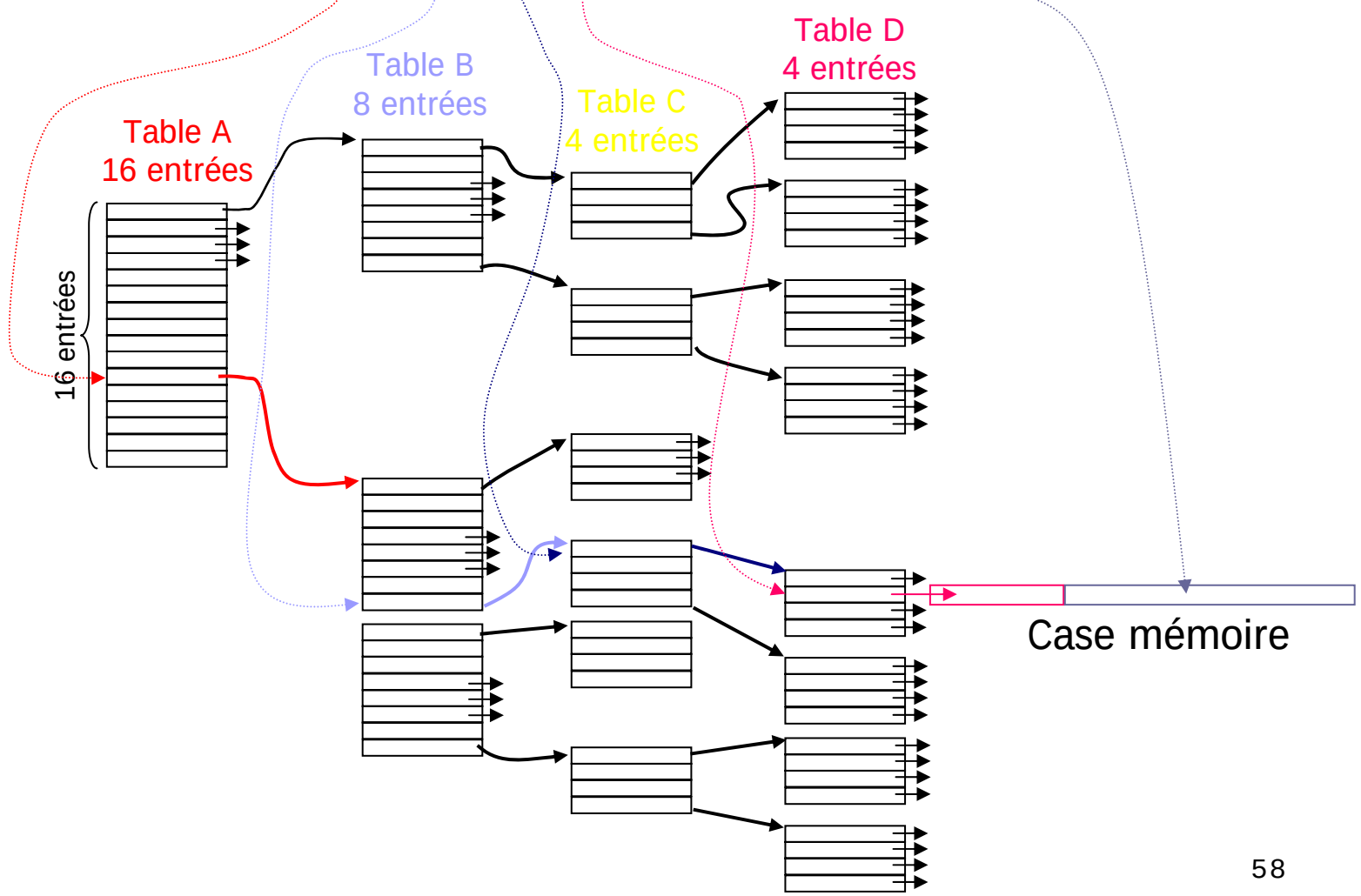


Le contexte indique au MMU l'index de la première table de haut niveau du processus en cours d'exécution.

Le 68030

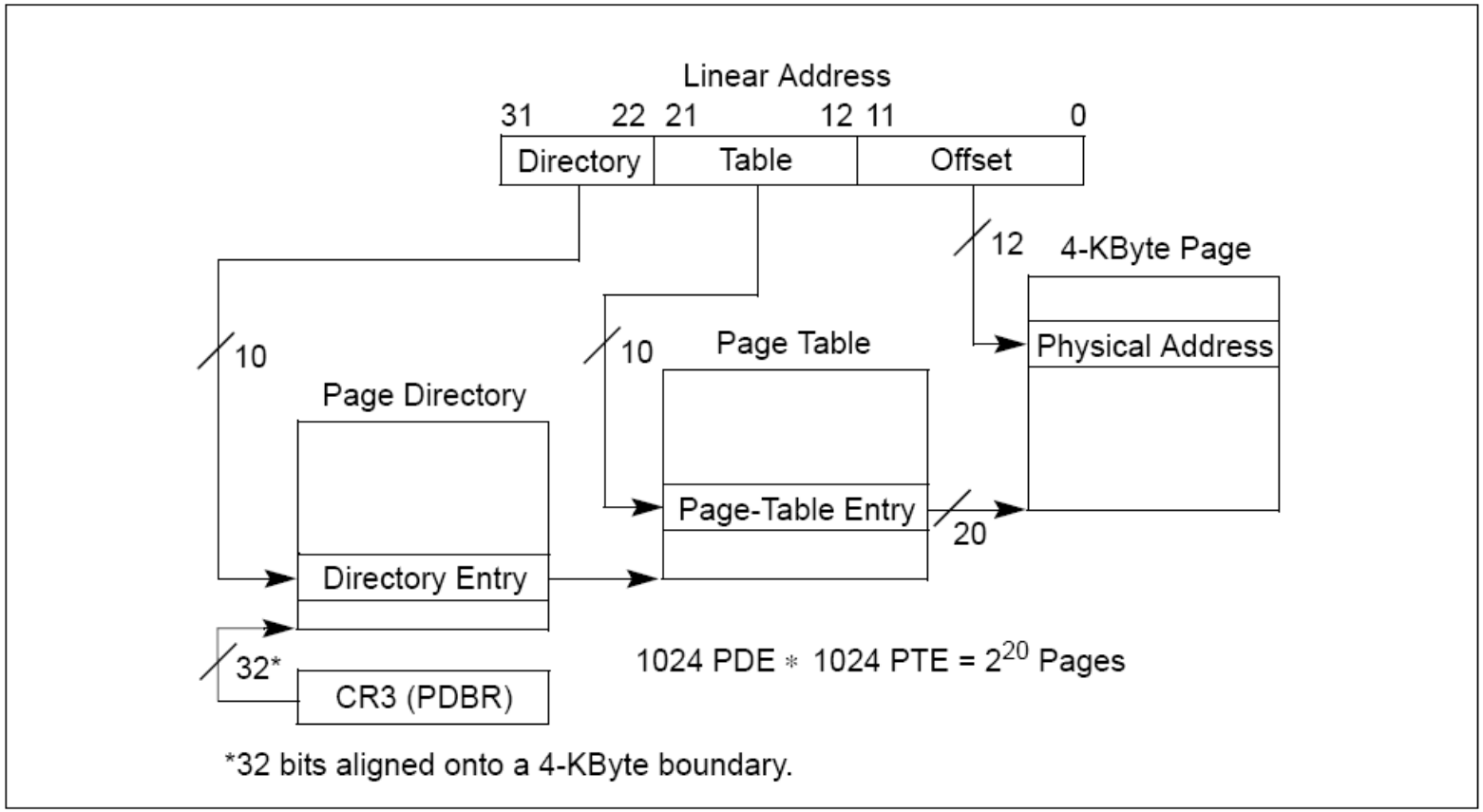
Adresse de 32bits

10	4	3	2	2	11
0000000000	1010	111	00	01	00000000110
	A	B	C	D	Dép



MMU Intel

Deux niveaux d'indirections : une adresse linéaire (adresse logique utilisée par le processeur), composée d'une entrée dans un répertoire de page, d'une entrée dans une table de page, et d'un déplacement dans cette page (possible d'adresser 2^{20} pages de 4 kilo-octets, soit 4Go de données).



TLB (*Translation Lookaside Buffer*)

C'est une mémoire associative qui mappe les adresses virtuelles sur les adresses physiques sans passer dans les tables de pagination, elle fait souvent partie du MMU et elle est petite, elle contient moins de 32 entrées.
Chaque entrée contient :

- Le numéro de page virtuelle.
- Un indicateur de modification de la page.
- Un code de protection.
- Le numéro de la case mémoire physique.

Ces champs correspondent un à un aux champs de la table des pages.

Indicateur de validité
Pages virtuelles
Indicateur de modification
Indicateur de protection
Case mémoire

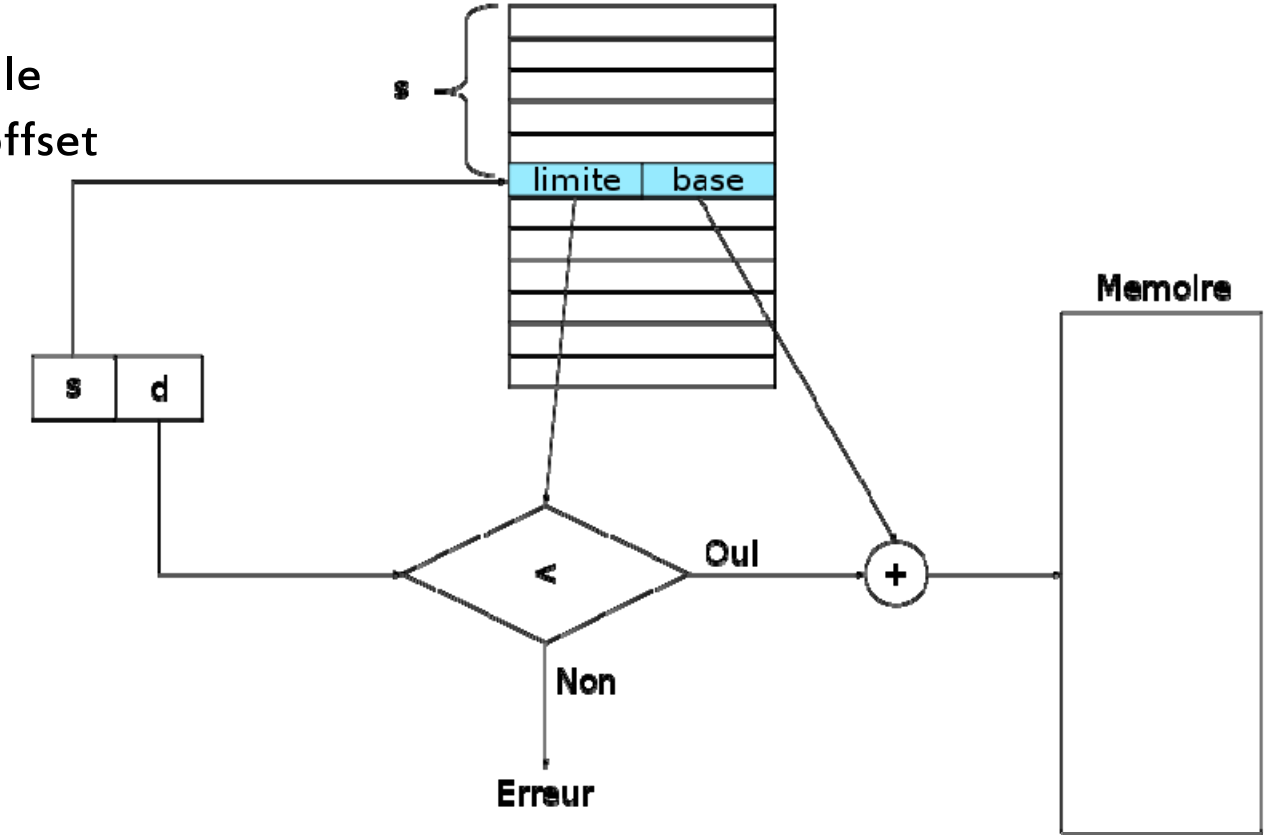
1	140	1	rw	31
1	20	0	rx	39
1	130	1	rw	29
1	129	1	rw	62
1	19	0	rx	50
1	21	0	rx	45
1	860	1	rw	14
1	861	1	rw	75

La segmentation

Un segment est un ensemble d'emplacements mémoire consécutifs non sécable. La mémoire apparaît comme un espace à deux dimensions. Le compilateur définit ces segments (par exemple: un segment de code, un segment de données, un segment de pile).

Segment = base + taille

Adresse = n°segm. + offset



Segmentation

La segmentation a été créé pour :

- Avoir des dimensions de segments variables.
- Répartir les données et le code dans des segments différents.
- Simplifier le partage de procédure entre processus.
- Offrir une protection au contenu du segment.
- Offrir un niveau de privilège au contenu du segment

Les solutions mixtes

La pagination segmentée, où la table des pages sera segmentée. Autrement dit, le numéro de page p du couple (p, d) de l'adresse virtuelle sera interprété comme un segment (s, p'). Ce système résout le problème de taille de la table des pages.

La segmentation paginée, où chaque segment sera paginé. Autrement dit, le champ déplacement d du couple (s, d) de l'adresse virtuelle sera interprété comme un numéro de page et un déplacement (p, d').

Pagination - segmentation

Considérations	Pagination	Segmentation
Le programmeur doit-il connaître la technique utilisée ?	Non	Oui
Combien y a-t-il d'espaces d'adressage linéaire ?	1	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Oui	Oui
Les procédures et les données peuvent elles être distinguées et protégées séparément ?	Non	Oui
Peut on traiter simplement des tables dont les tailles varient ?	Non	Oui
Le partage de procédure entre les utilisateurs est il simplifié ?	Non	Oui
Pourquoi cette technique a-t-elle été inventée ?	Pour obtenir un grand espace d'adressage linéaire	Pour permettre la séparation des programmes et des données dans des espaces d'adressage indépendants et pour faciliter le partage et la protection

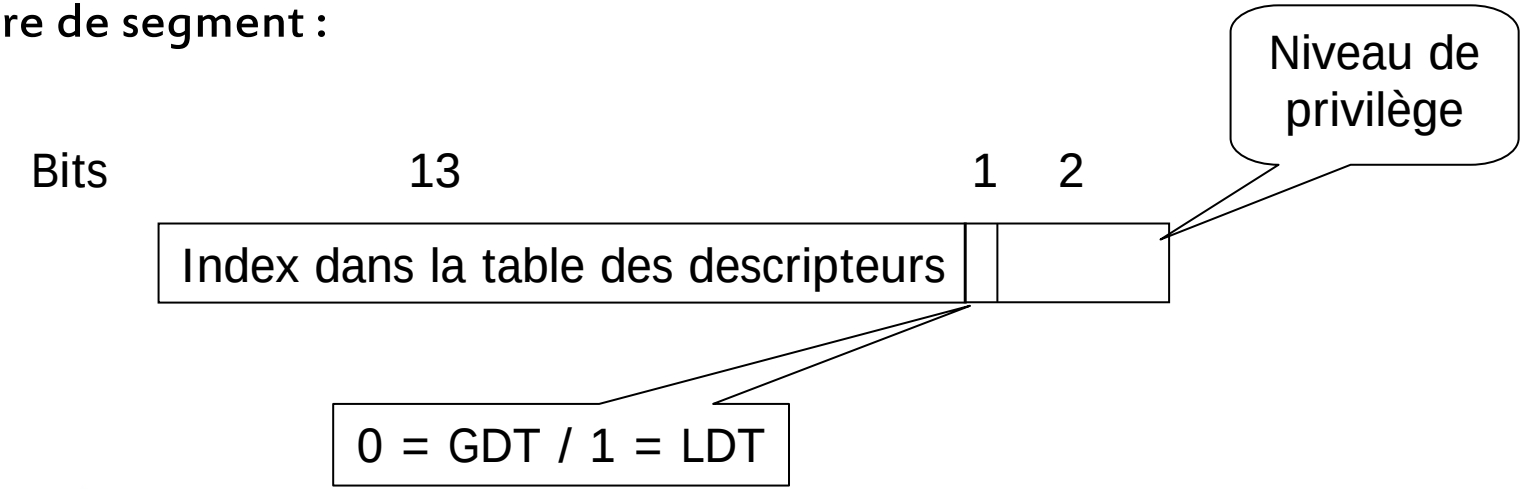
Exemple de segmentation avec pagination : 80386

Chaque segment est définit par un descripteur de segment (contient, par exemple, l'adresse de début de segment, la granularité, la limite)

La table des descripteurs locaux : LDT (Local Descriptor Table) est unique pour chaque processus.

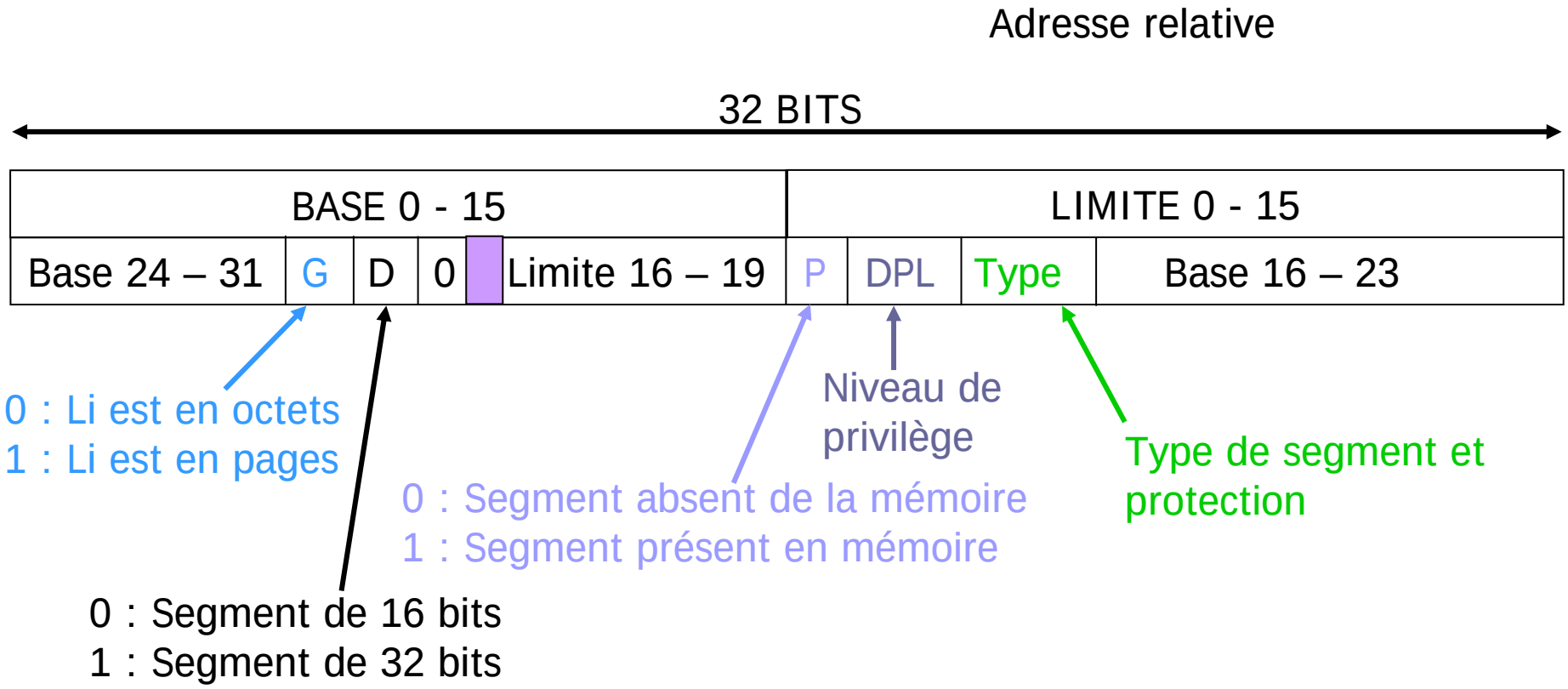
La table des descripteurs globaux GDT (Global Descriptor Table) est commune au système.

Le 80386 accède à un segment à travers son sélecteur en le chargeant dans un registre de segment :



Exemple de segmentation avec pagination : 80386

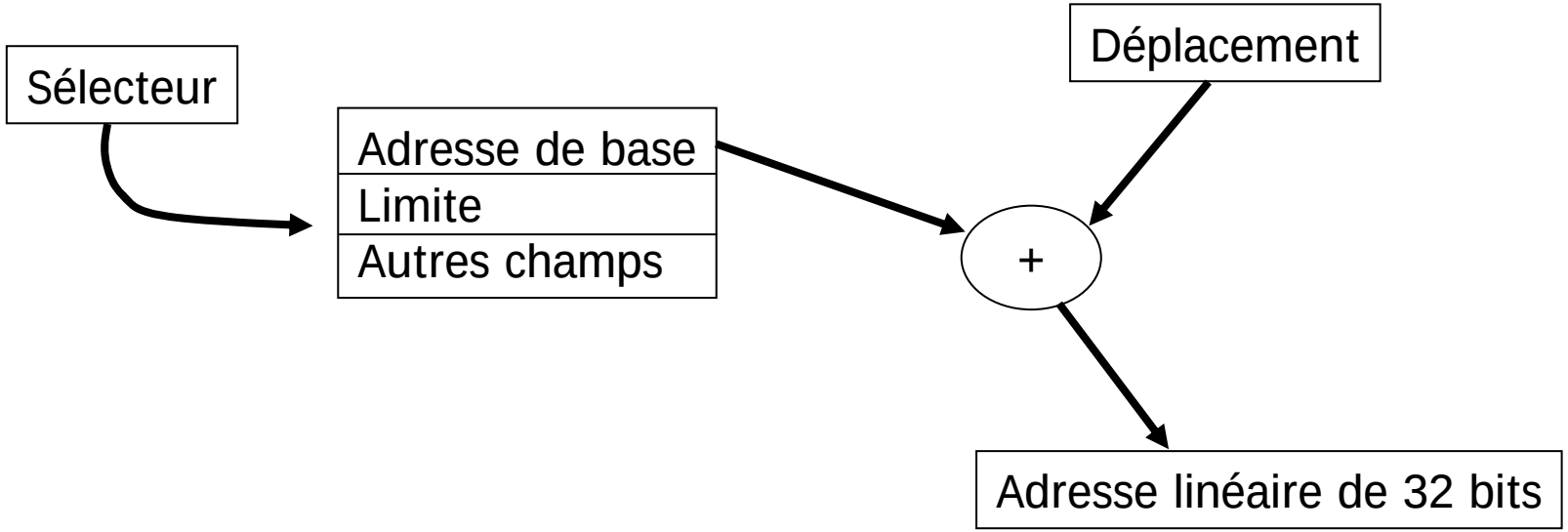
Le chargement du sélecteur entraîne le chargement du descripteur (contenu dans la table) dans un registre du 80386



Exemple de segmentation avec pagination : 80386

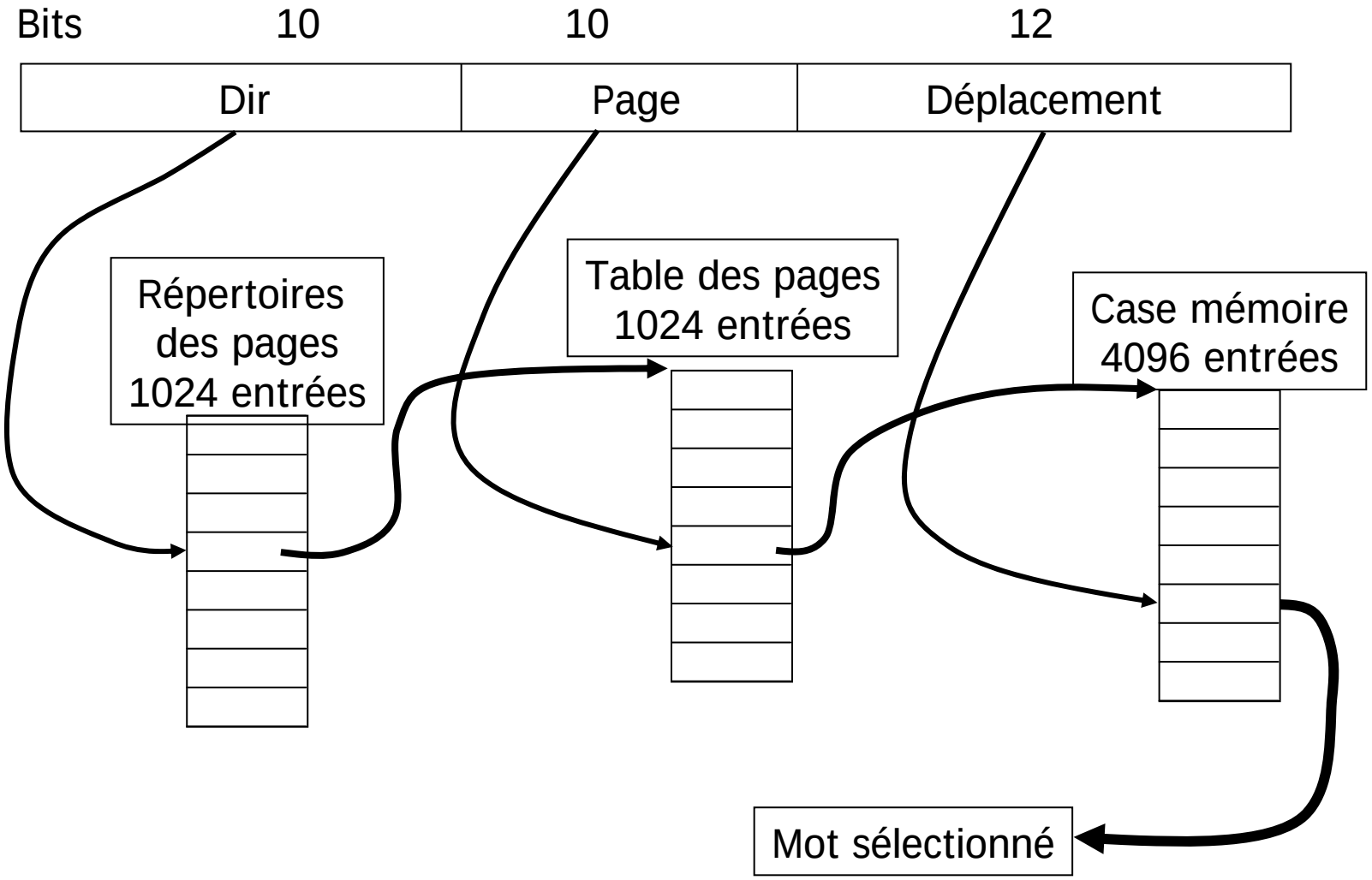
(Sélecteur, déplacement) :

- Trouver le descripteur du segment répertorié par le sélecteur
- Vérifier si le déplacement dépasse la fin du segment (champ limite)
- Ajouter les 32 bits de base au déplacement de façon à former l'adresse linéaire



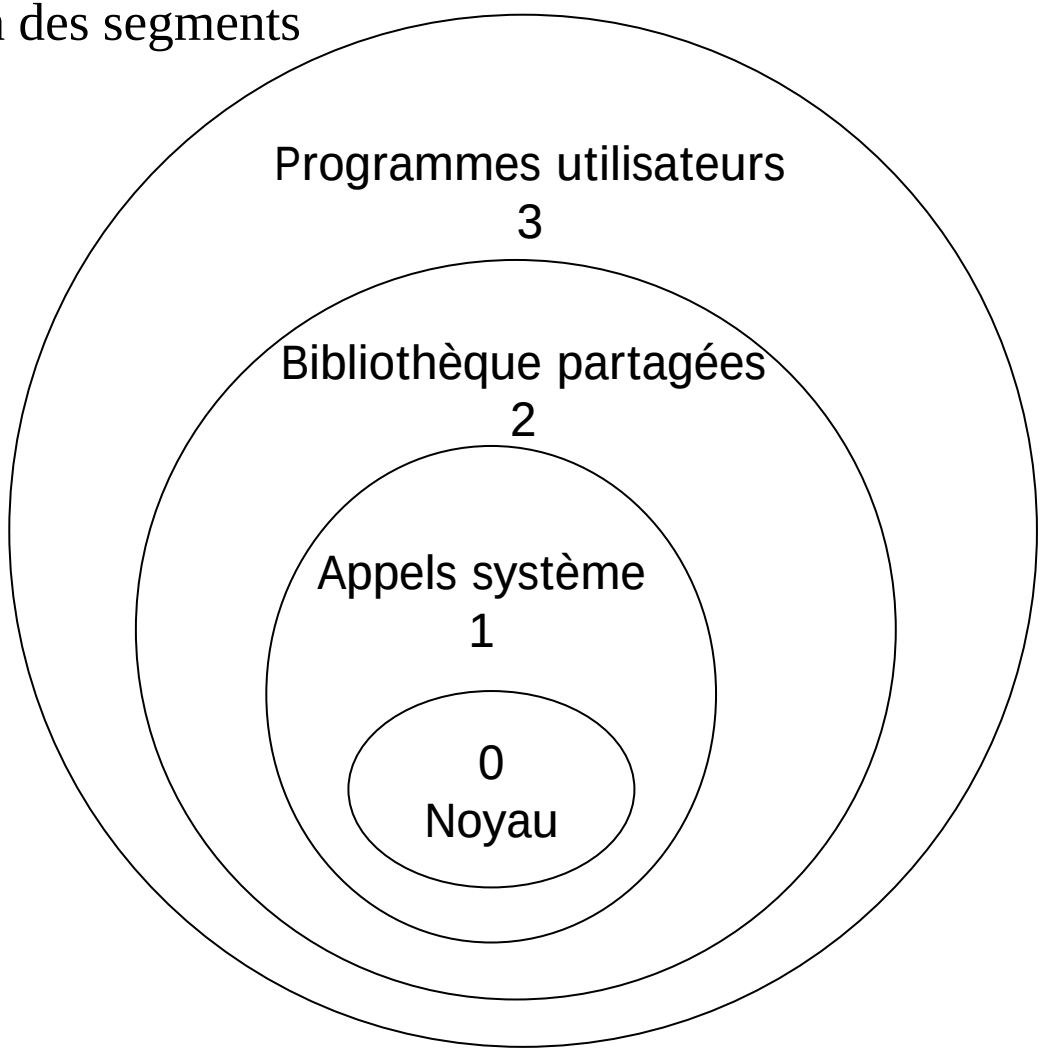
S'il n'y a pas de pagination, l'adresse linéaire est l'adresse physique
Si la pagination est activée, l'adresse linéaire représente l'adresse virtuelle à deux niveaux

Exemple de segmentation avec pagination : 80386



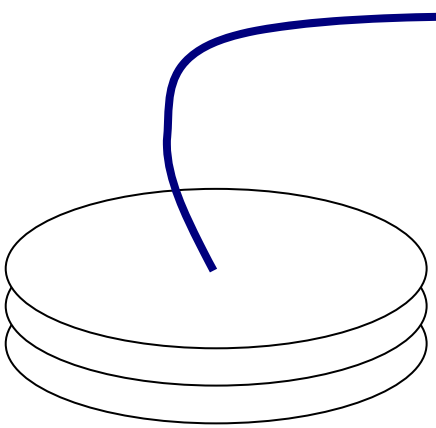
Exemple de segmentation avec pagination : 80386

Exemple de protection des segments



Exemple

Au démarrage du système

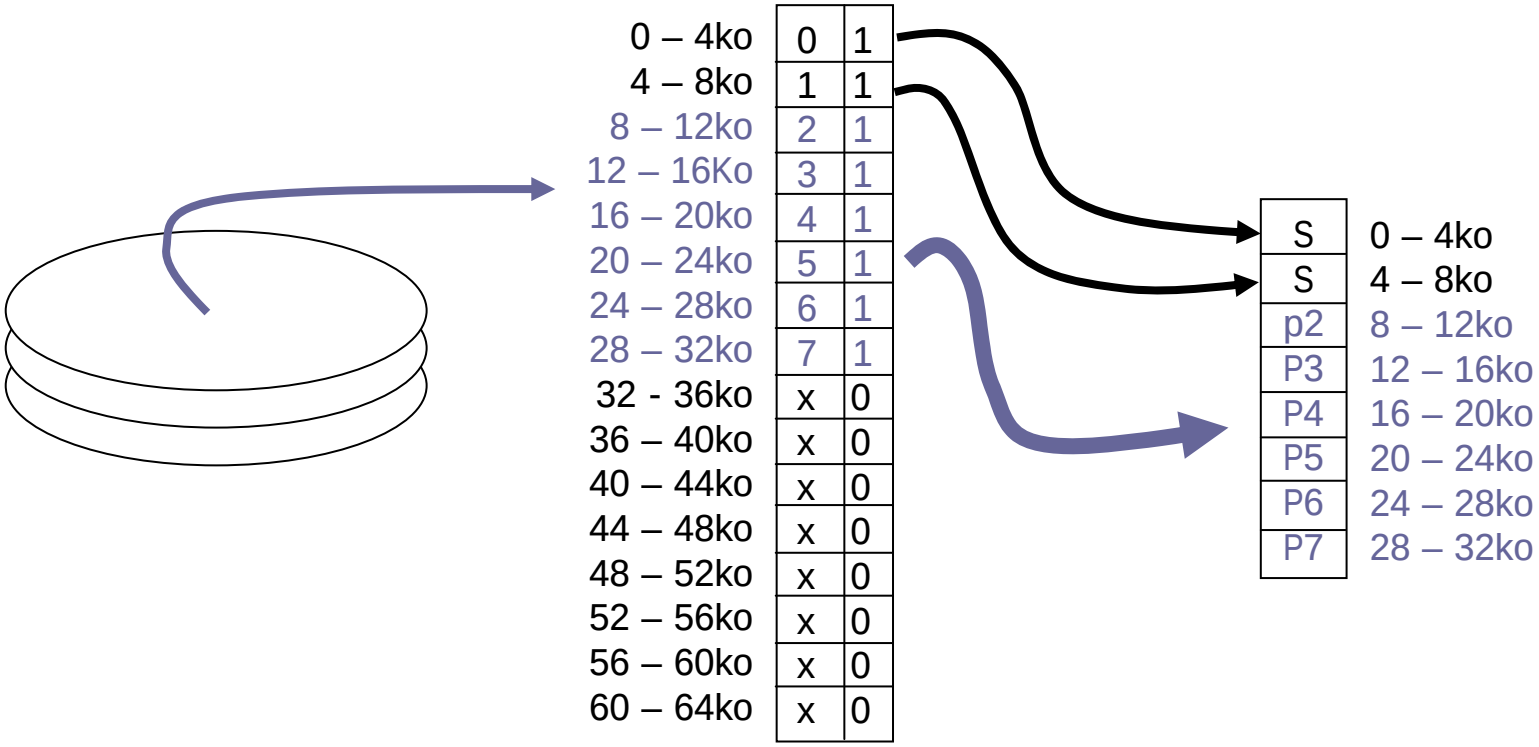


0 – 4ko	0	1
4 – 8ko	1	1
8 – 12ko	x	0
12 – 16Ko	x	0
16 – 20ko	x	0
20 – 24ko	x	0
24 – 28ko	x	0
28 – 32ko	x	0
32 - 36ko	x	0
36 – 40ko	x	0
40 – 44ko	x	0
44 – 48ko	x	0
48 – 52ko	x	0
52 – 56ko	x	0
56 – 60ko	x	0
60 – 64ko	x	0

S	0 – 4ko
S	4 – 8ko
x	8 – 12ko
x	12 – 16ko
x	16 – 20ko
x	20 – 24ko
x	24 – 28ko
x	28 – 32ko

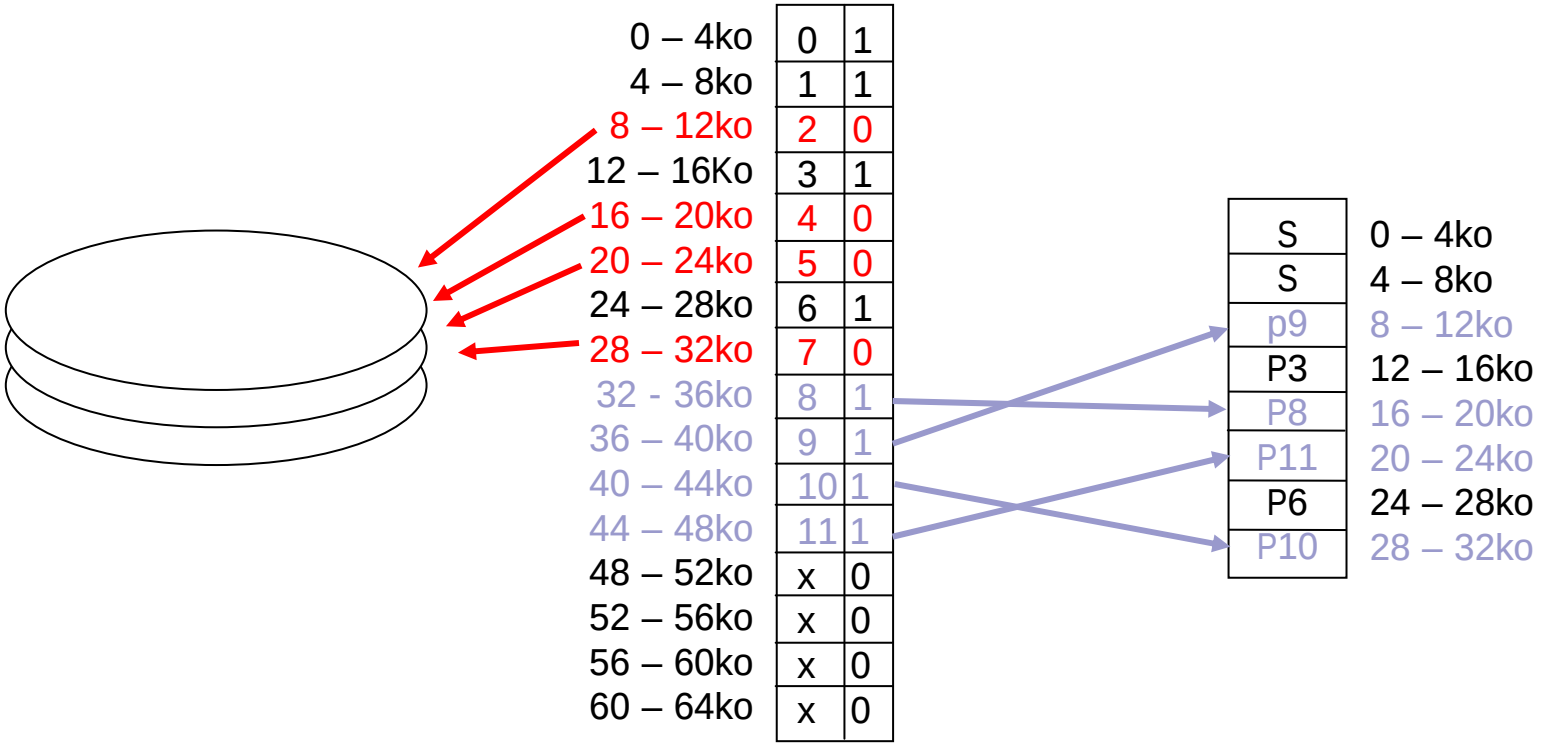
Exemple

Après lancement d'un processus consommant de la mémoire



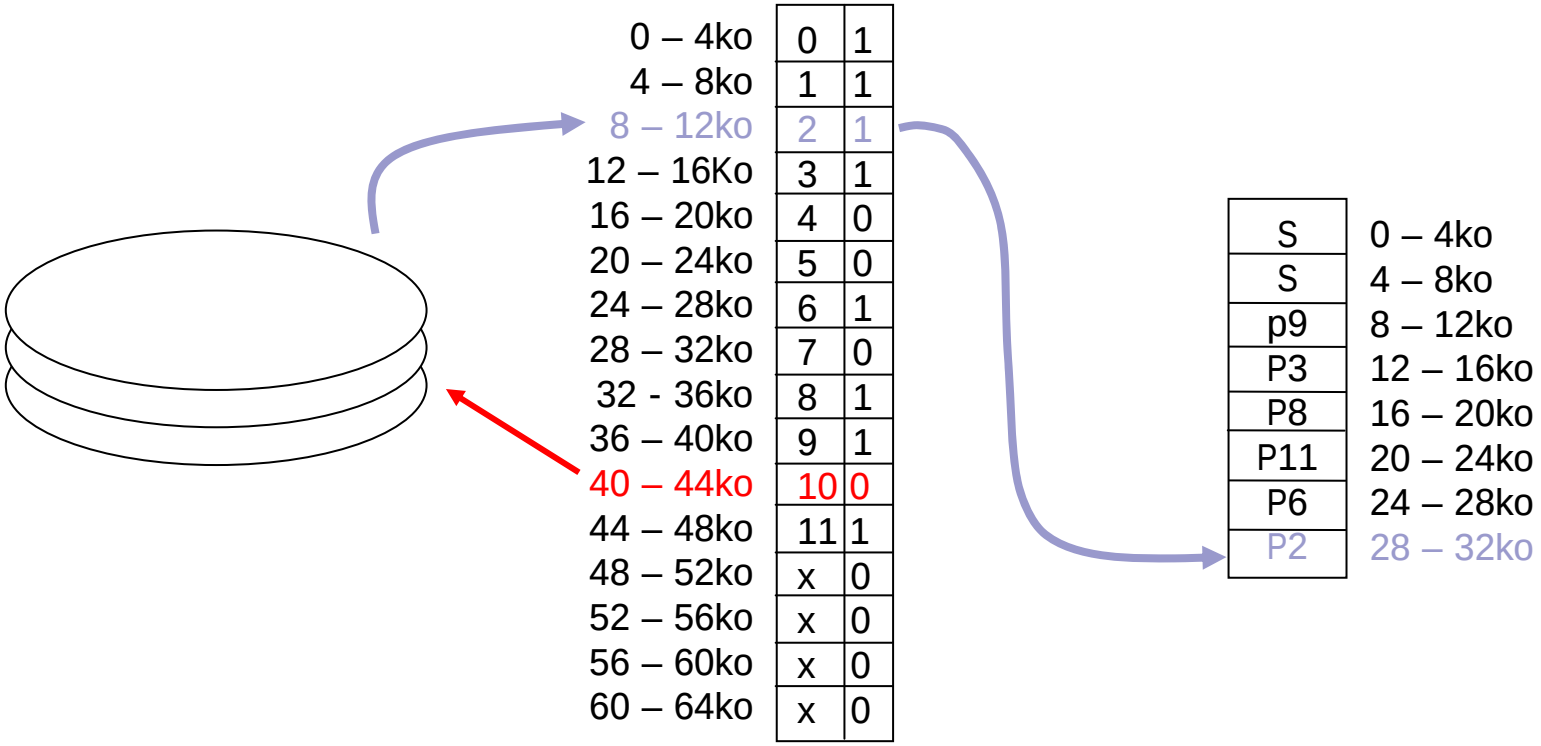
Exemple

Après quelques secondes d'exécution, le processus demande 16ko de mémoire



Exemple

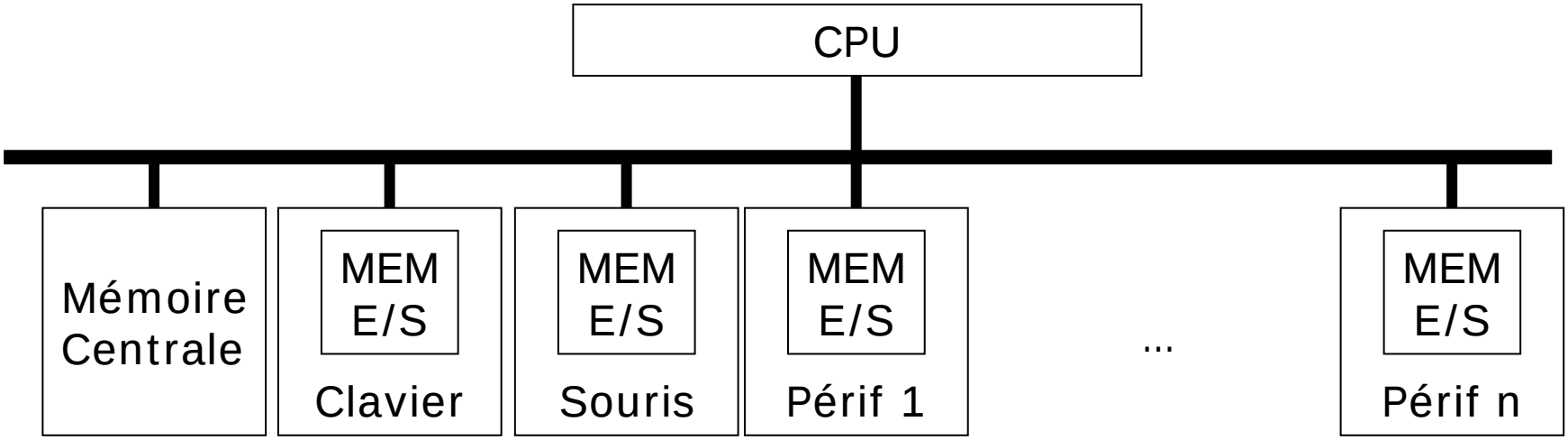
Puis le processus fait appel à une adresse virtuelle de la page 8 – 12ko



Dialogue avec le matériel

Les accès au matériel se font par l'intermédiaire d'une mémoire d'entrées – sorties. Cette mémoire est connectée physiquement au matériel et se trouve (la plupart du temps) sur les cartes périphériques.

Pour piloter une carte périphérique, les programmes écrivent dans cette mémoire.



Dialogue avec le matériel

Lorsque le matériel a été programmé pour effectuer un travail, il signale la fin de son travail :

- Par une interruption.
- Par une réponse à une demande logicielle.

La première méthode est appelée mécanisme d'interruption et la seconde mécanisme de scrutation.

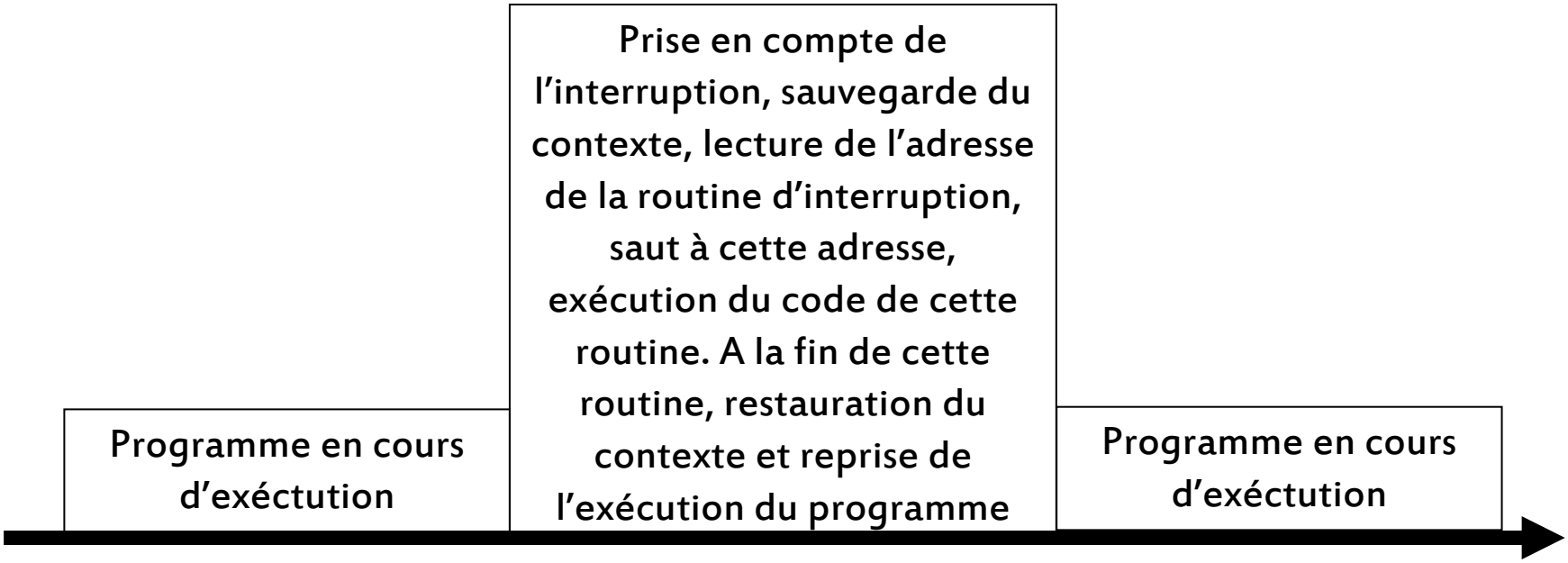
Le mécanisme d'interruption est le plus utilisé aujourd'hui.
Le mécanisme de scrutation est surtout utilisé dans l'informatique industrielle.

Mécanisme d'interruption

Ce mécanisme permet d'interrompre un programme en cours d'exécution:

1. Au moment de l'interruption, le processeur reçoit sur une patte un signal électrique, il passe automatiquement en mode non interruptible.
2. A la réception de ce signal, il sauvegarde l'ensemble de ses registres et toutes informations nécessaires à la reprise du programme en cours d'exécution.
3. Il lit ensuite l'identité de l'interruption par un dialogue sommaire avec le composant contrôlant le mécanisme d'interruption.
4. Il calcule l'adresse du vecteur d'interruption lié à cette identité d'interruption.
5. Il charge depuis cette adresse une adresse pointant sur une routine appelée routine d'interruption.
6. Il effectue un saut à cette adresse pour exécuter cette routine, souvent la première instruction de cette routine est de ré – autoriser les interruptions.
7. Il exécute ce code jusqu'à la fin de la routine.
8. Il restaure l'état précédent l'interruption.
9. Il reprend le cours du programme qui était en cours d'exécution.

Mécanisme d'interruption



Mécanisme d'interruption

Il existe plusieurs interruptions dans un système, elles sont classées par une priorité liée à leurs connexions physiques ou programmées au démarrage du système d'exploitation.

Par exemple les PC disposent de 16 niveaux d'interruptions, leur contrôleur est le circuit 8259 d'Intel permettant de gérer 8 niveaux :

0	Horloge système
1	Clavier
2	IRQ 8 à 15 (redirection)
3	Port série COM2
4	Port série COM1
5	Port parallèle LPT2
6	Contrôleur de disquette
7	Port parallèle LPT1
8	Horloge temps réel
9	Disponible
10	Disponible
11	Disponible sauf scsi
12	Disponible
13	Co – processeur math
14	Contrôleur de disque IDE
15	Contrôleur de disque E-IDE

Une interruption est interruptible par une interruption de plus haute priorité mais pas par une interruption de priorité inférieure.

Dans le cas des PC les priorités sont dans l'ordre décroissant.

Mécanisme de scrutation

Le mécanisme de scrutation consiste à interroger d'une façon répétitive le périphérique.

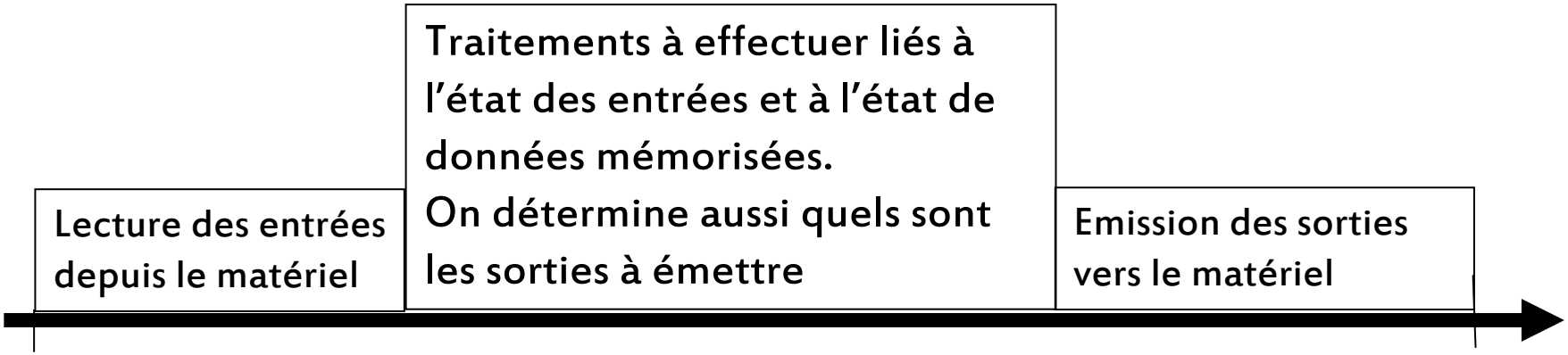
Pour connaître l'état d'une imprimante, un programme doit donc boucler de la façon suivante :

1)As-tu fini d'imprimer ?
2)Petit travail utile
3)Retour en 1.

Cependant c'est un mécanisme très utilisé dans l'informatique industrielle ou les temps de cycle sont connus. Le matériel est adapté pour produire des résultats en un temps donné, par exemple 50 ms. Le cycle est alors de 50 ms.

Le programme doit à un top (0 ms) lire les entrées, les traiter puis émettre les nouvelles commandes avant le top suivant (à 50 ms).

Mécanisme de scrutation



Ce type de système est surtout destiné à l'embarqué et dispose d'un système d'exploitation très rudimentaire.

En général, les cartes électroniques sur lesquelles le logiciel s'exécute sont développées simultanément au logiciel.

Exemple de système : Météor (Ratp – Matra – Altsom).

Le DMA

Les entrées – sorties gérées par interruptions libèrent le processeur, mais :

Le processeur passe beaucoup de temps à effectuer des instructions de transfert de données.

Beaucoup d'événements d'entrées – sorties nécessitent des transferts de blocs

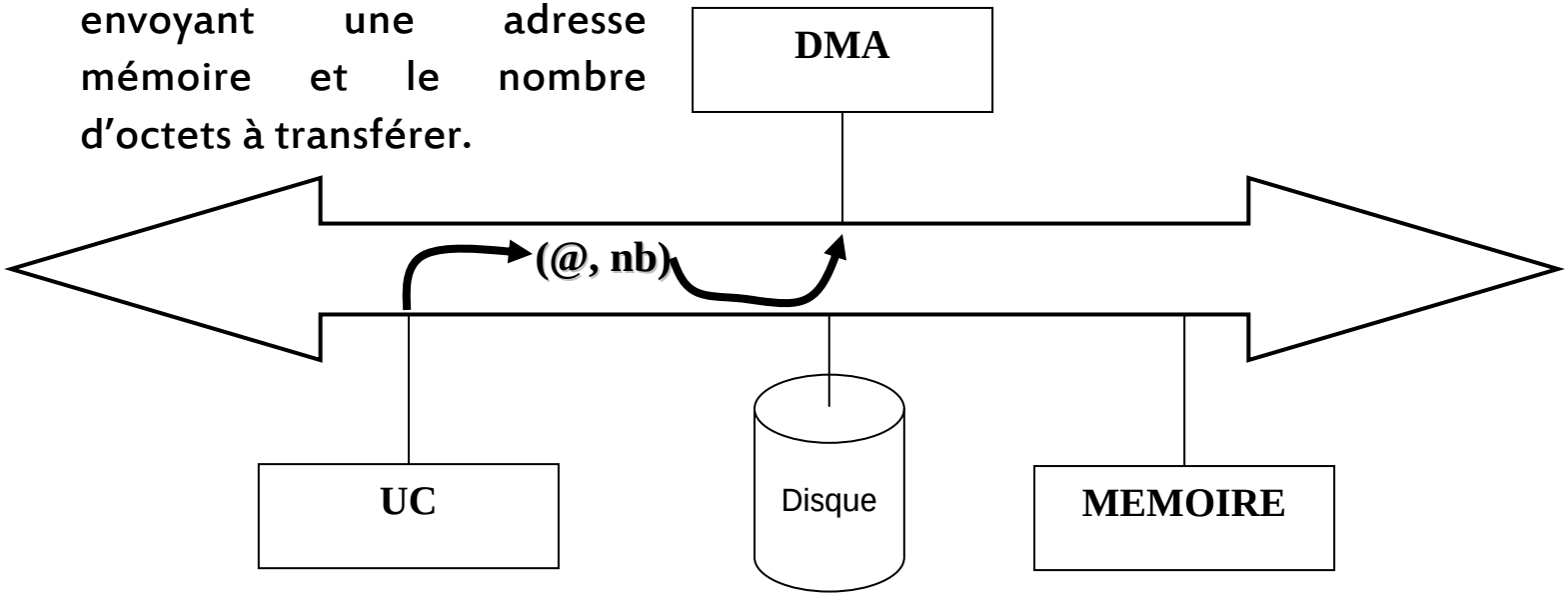
La plupart des ordinateurs possèdent un processeur spécialisé qui gère le dispositif d'accès à la mémoire :

LE DMA

(Direct Memory Access : accès direct à la mémoire).

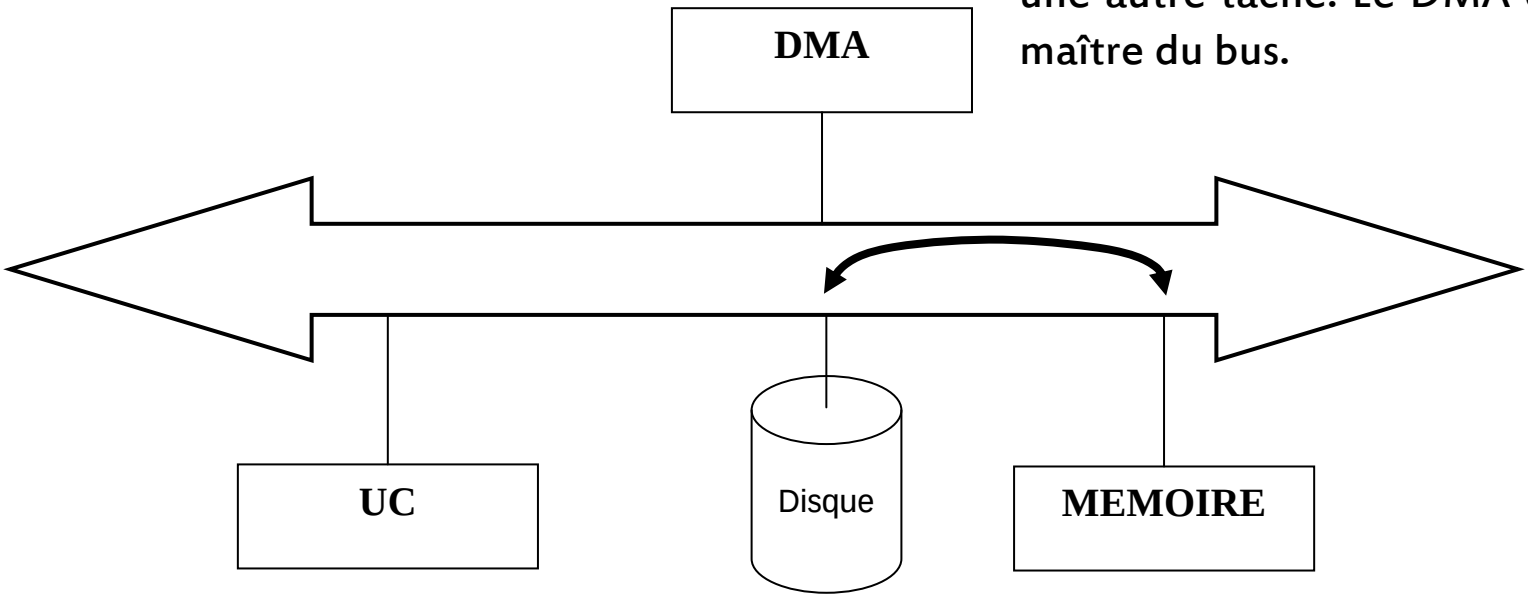
Le DMA

❶ Le processeur initialise les registres du DMA en lui envoyant une adresse mémoire et le nombre d'octets à transférer.



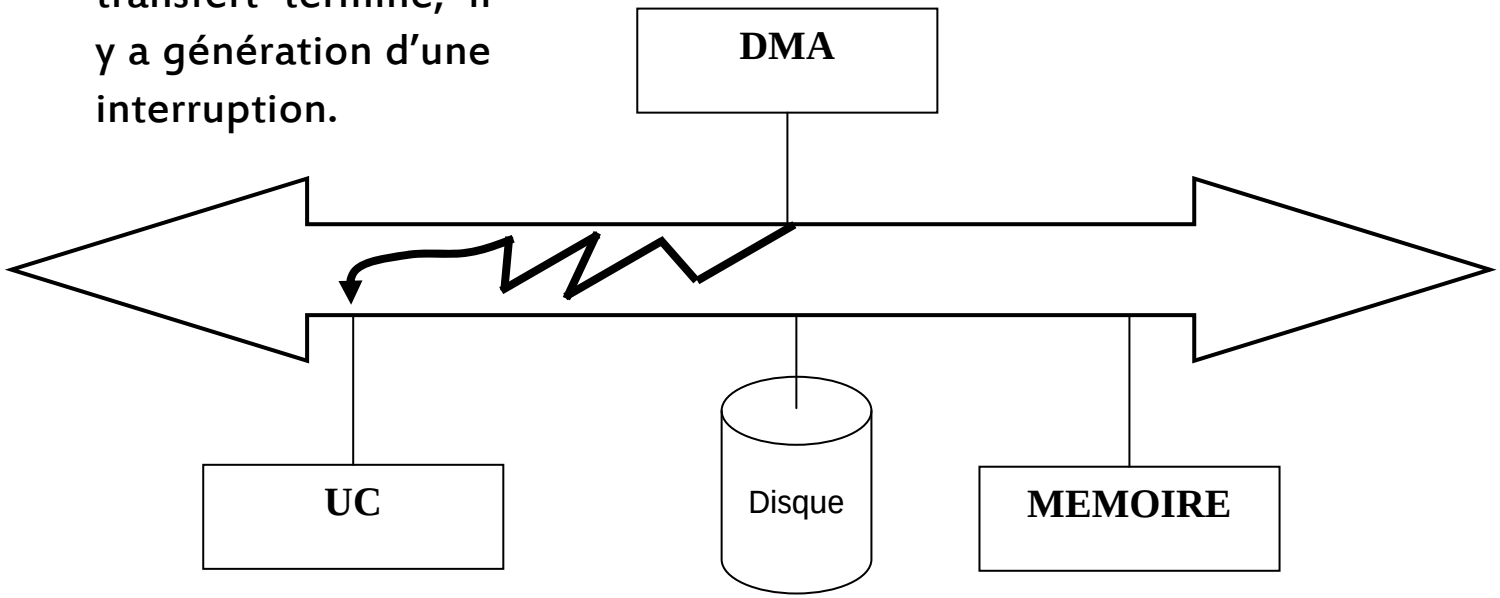
Le DMA

❷ Le DMA transfert les données entre un disque et la mémoire, pendant ce temps, l'UC travaille a une autre tache. Le DMA doit être maître du bus.



Le DMA

❸ Une fois le transfert terminé, il y a génération d'une interruption.



Le DMA

Il peut y avoir plusieurs DMA dans un même ordinateur :
Il faut dans ce cas avoir un arbitre pour accéder au bus.

Les DMA peuvent être plus ou moins intelligents, ils forment alors des processeurs d'entrées – sorties :

Ces processeurs opèrent à partir de programmes fixes, ou de programmes chargés par le système d'exploitation.

Par exemple : l'imprimante LP11 provoqueraient 4800 interruptions pour imprimer une page de 60 lignes de 80 caractères, un processeur d'entrées – sorties pourraient éviter 4799 interruptions.

Puisque nous parlons de processeurs d'entrées – sorties capables d'exécuter des instructions, nous sommes presque en train de parler de multiprocesseurs.

Toutefois ces processeurs n'ont pas pour vocation d'effectuer que des transferts de données et non des instructions générales.

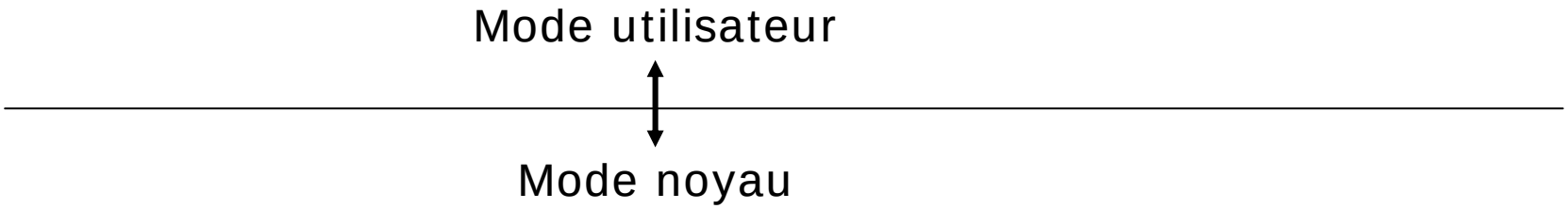
Notion Système

Les systèmes d'exploitation d'aujourd'hui possèdent deux modes de fonctionnement :

- Mode utilisateur ou applicatif
- Mode noyau (kernel)

Le mode utilisateur permet aux applications de s'exécuter. Ces applications n'accèdent aux ressources du système, y compris et surtout celles matérielles, qu'en utilisant « les services » du noyau du système.

Le noyau du système garantit l'intégrité de fonctionnement, évitant les plantages dus aux applications mal programmées.



Les pilotes

Le dialogue entre les deux modes est effectué par des appels système. Lorsqu'une application veut utiliser un périphérique, elle se sert d'un pilote de périphérique lié au périphérique considéré.

Ce pilote contient des services dont certains sont proposés aux applications. Par exemple pour un driver d'impression, il est possible d'implémenter les services suivants :

-Chargement et déchargement du pilote (par une commande ou au démarrage/arrêt).
-Traitements liés aux interruptions générées par l'imprimante.
-Impression d'un document.
-Annulation d'une impression.
-Commande pour utiliser un mode l'imprimante, par exemple graphique/texte.

Les interruptions du pilote avertissent l'application d'un événement lié à l'imprimante par l'intermédiaire d'objets de synchronisation ou de communication. A l'application alors d'aller interroger le pilote pour connaître le résultat de sa commande initiale.

Les pilotes

Le pilote de périphérique contient au moins les services suivants :

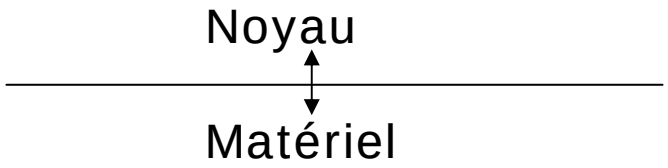
Chargement du pilote de périphérique dans le noyau manuellement ou automatiquement au démarrage.
Déchargement du pilote dans le noyau manuellement ou automatiquement à l'arrêt.
Ouverture du pilote de périphérique (vu comme un fichier depuis une application).
Fermeture du pilote de périphérique.
Ecriture de commandes dans le pilote de périphérique.
Lecture du résultat de commandes depuis le pilote de périphérique.
Routines d'interruptions liés au périphérique considéré.

A cela peut s'ajouter des routines de contrôle offrant, par l'intermédiaire de messages connus de l'application, des services autres que ceux ci-dessus tels que :

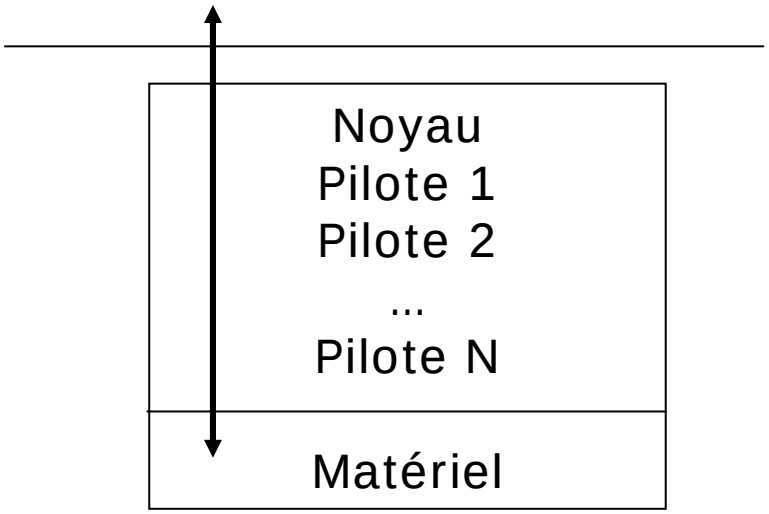
Annulation de la dernière commande
Ré – initialisation du périphérique
...

Les pilotes multicouches

Un pilote adresse du matériel :



Mais aussi d'autres pilotes, formant ainsi des pilotes virtuels disposés en couches :



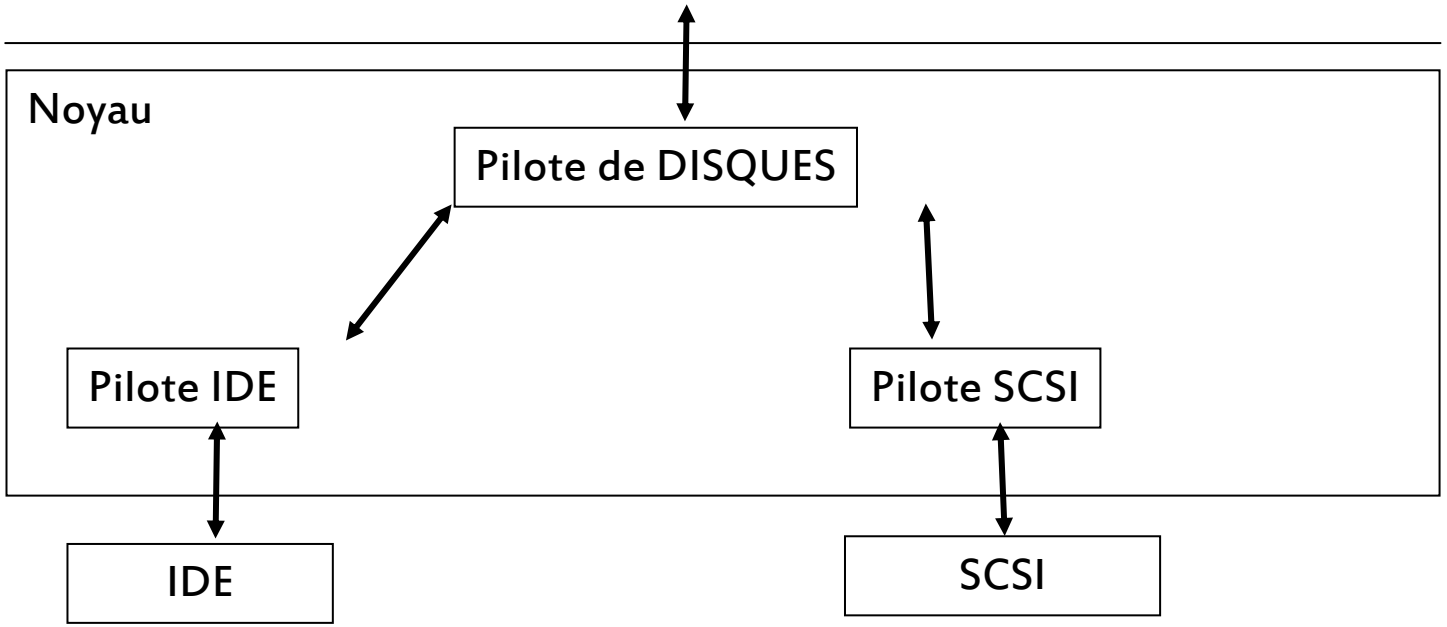
L'ensemble de ces pilotes permettent d'abstraire certains périphériques.

Exemple de pilote multicouche

Supposons qu'un ordinateur dispose de disques IDE et SCSI.

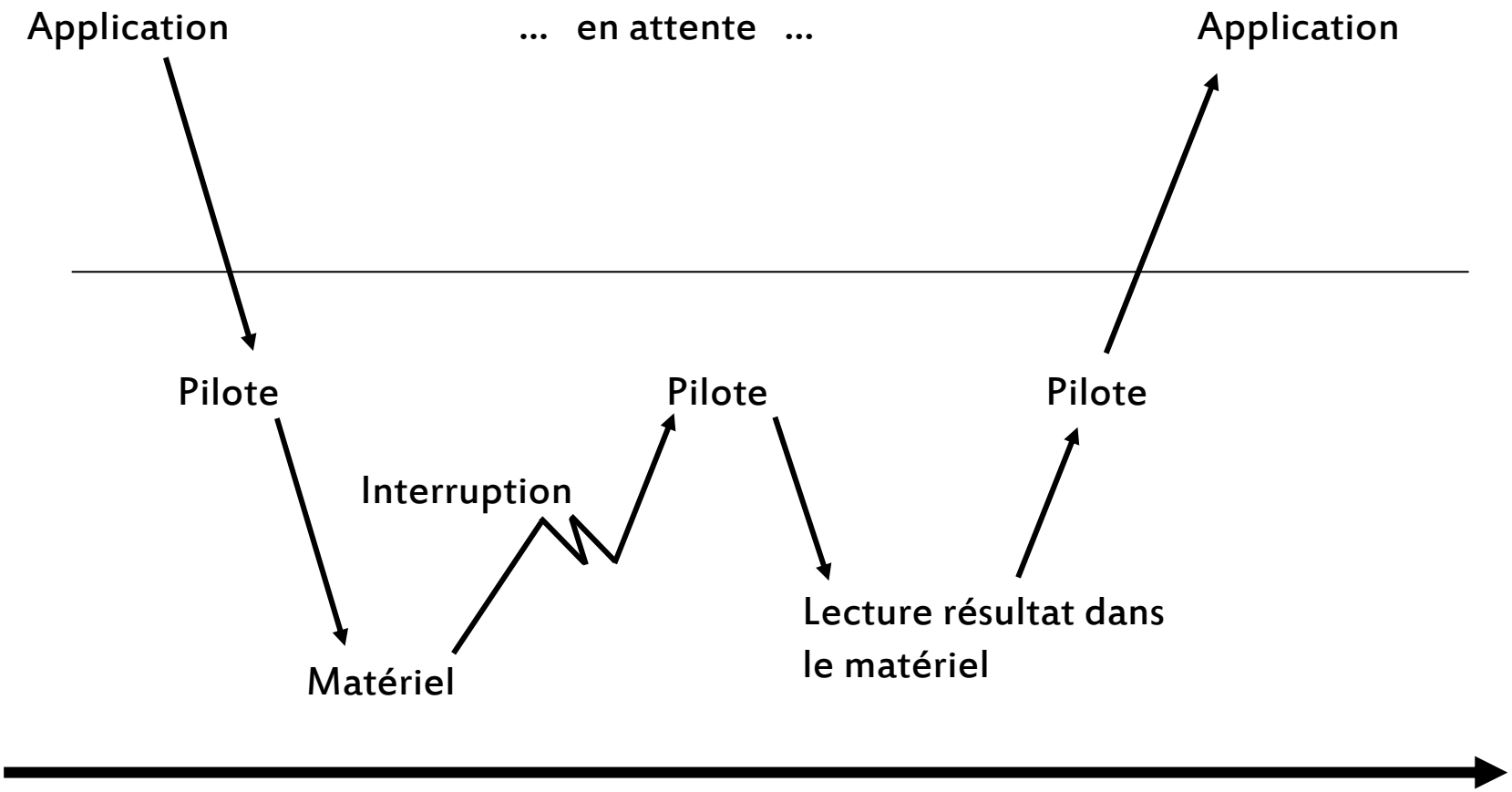
Lorsqu'une application écrit dans un fichier, elle se sert des services d'un pilote multi couches contrôlant l'écriture sur le disque.

Ce pilote identifie le type de disque et effectue le travail demandé par l'intermédiaire d'un autre pilote lié au type de disques.



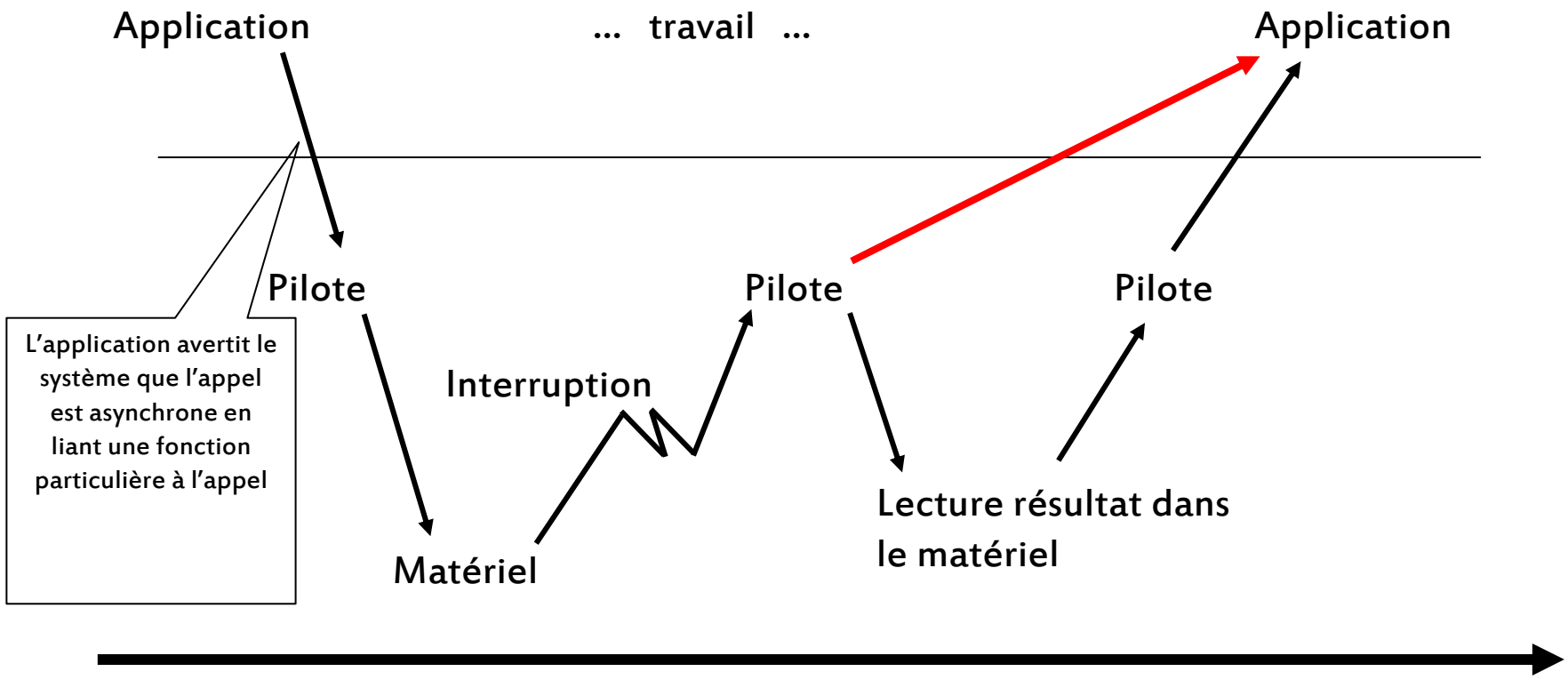
Les pilotes synchrones

Lorsqu'une application doit attendre le résultat d'un travail demandé à un périphérique, on dit que le pilote est synchrone :



Les pilotes asynchrones

Lorsqu'une application n'attend pas le résultat d'un travail demandé à un périphérique, on dit que le pilote est asynchrone. L'application continue donc son travail et est avertie du résultat du service demandé par une routine spéciale de l'application liée lors de l'appel système.



Pour réaliser cela, l'application se sert d'une routine (APC asynchronous procedure call sous windows, un masque de signaux sous unix) qui l'avertira et lira le résultat du travail demandé.