Qu'est ce qu'un système d'exploitation ?

Les systèmes d'exploitation

- **MACHINE VIRTUELLE OU ETENDUE :**

  Permettre aux programmes d'utiliser les composants matériel 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 y font appel :

  GERER LES RESSOURCES

Les systèmes d'exploitation

- 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.

Les systèmes d'exploitation

- 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.

Les systèmes d'exploitation

- Troisième génération, 1965 – 1980 : les circuits intégrés

  - Multiprogrammation et connexion en ligne de plusieurs centaines d'utilisateurs.

  - Naissance et mort de MULTICS.

Les systèmes d'exploitation

- Quatrième génération, 1990 – 200X : les ordinateurs personnels

  - Réseau d'ordinateurs :

    - Système d'exploitation intègre « 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.

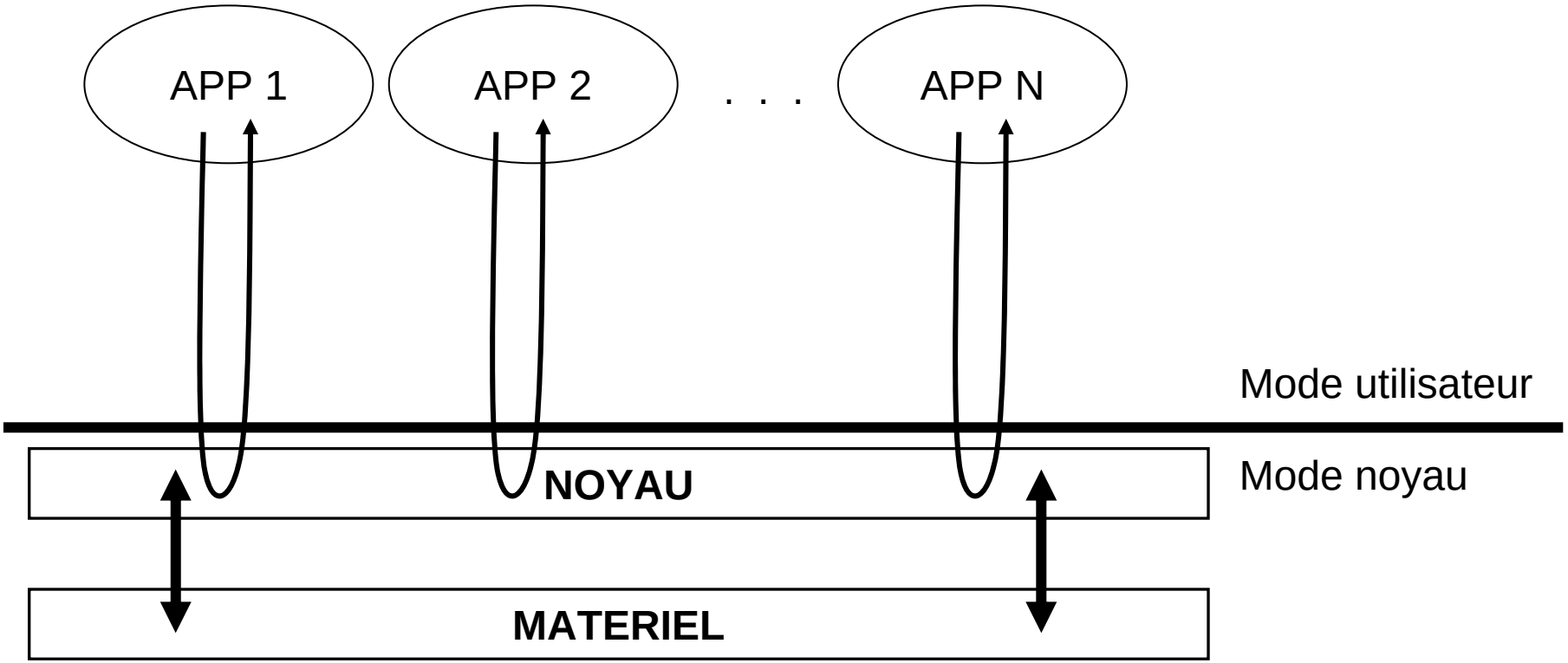
Les systèmes d'exploitation

- Les programmes communiquent avec le système par l'intermédiaire d'instructions étendues.

- Ces instructions sont appelées APPEL SYSTEME.

- Ces appels créent, utilisent et détruisent des objets qui sont gérés par le système.

MODE NOYAU – MODE UTILISATEUR



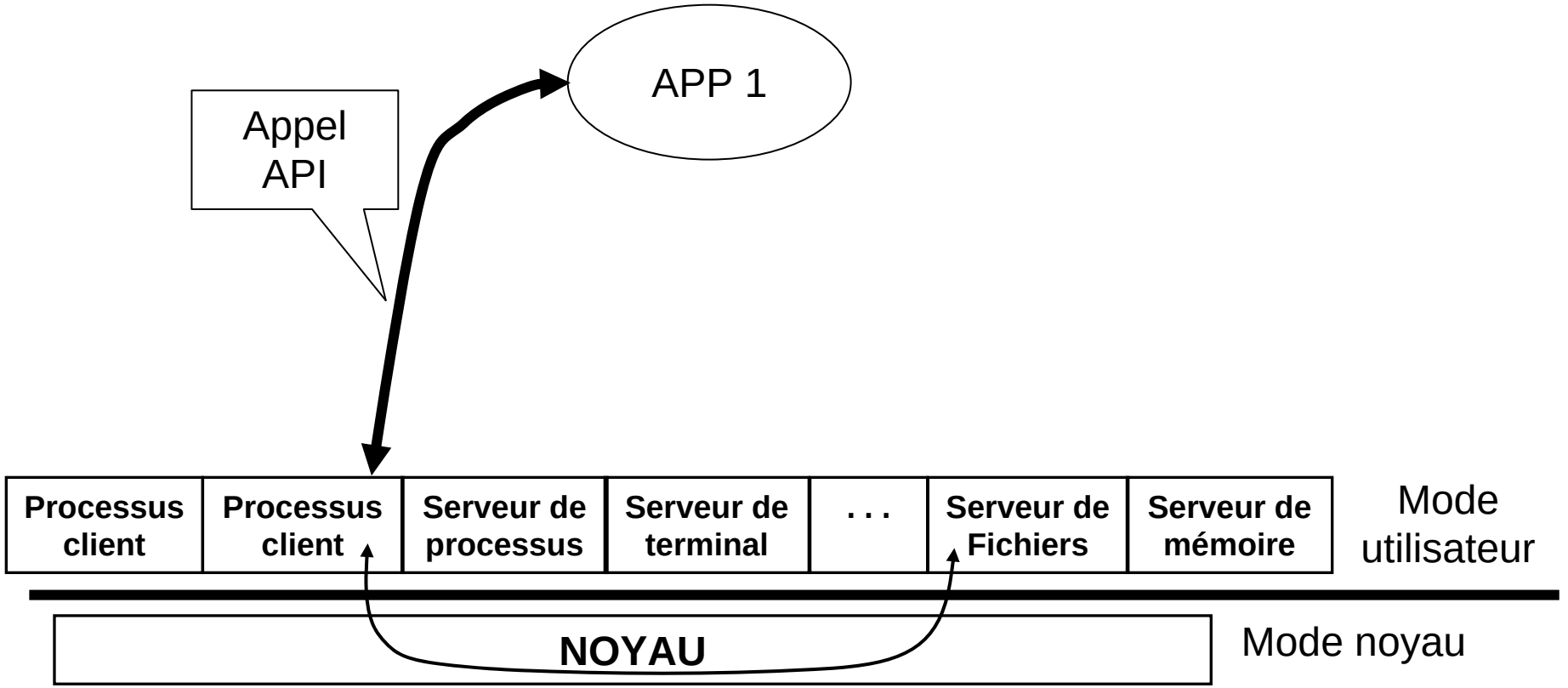
LE SYSTEME D'EXPLOITATION EST LE CODE QUI EXECUTE LES APPELS SYSTEME.

Les systèmes d'exploitation

- Les système monolithiques :
  - Absence totale de structures
  - Collection de procédures
    - Chacune peut appeler les autres à tout moment.
    - Chaque procédure est visible de toutes les autres.
    - Les appels système sont adressés en mettant les paramètres à des endroits bien définis.

Les systèmes d'exploitation

- Les mécanisme client – serveur :



Les systèmes d'exploitation

- Un système est un ensemble de programmes dont le but est de faciliter l'utilisation du matériel.

- Mais aussi d'assurer la gestion des ressources par l'utilisation d'objets gérés par le système.

- Les systèmes peuvent être structurés de différentes façons

Les systèmes d'exploitation

Entités d'exécution

Pour l'utilisateur une tâche c'est :

- Une application
- Un ensemble d'applications

Les systèmes d'exploitation
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 :

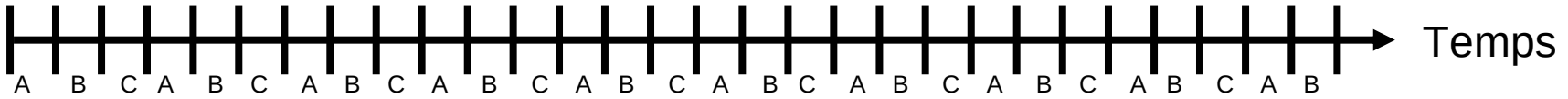
  $\Rightarrow$LES PROCESSUS

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

Les systèmes d'exploitation
Entités d'exécution

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



Les systèmes d'exploitation
Entités d'exécution

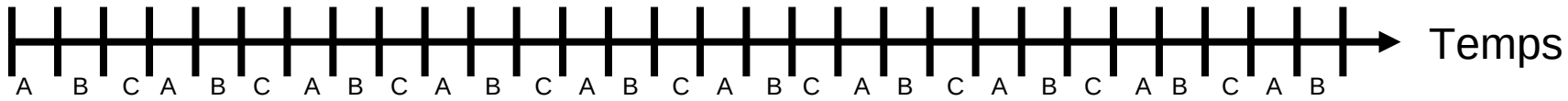
Les processus :

- Identité.
- Code exécutable.
- Une zone de mémoire propre au processus.
- 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.

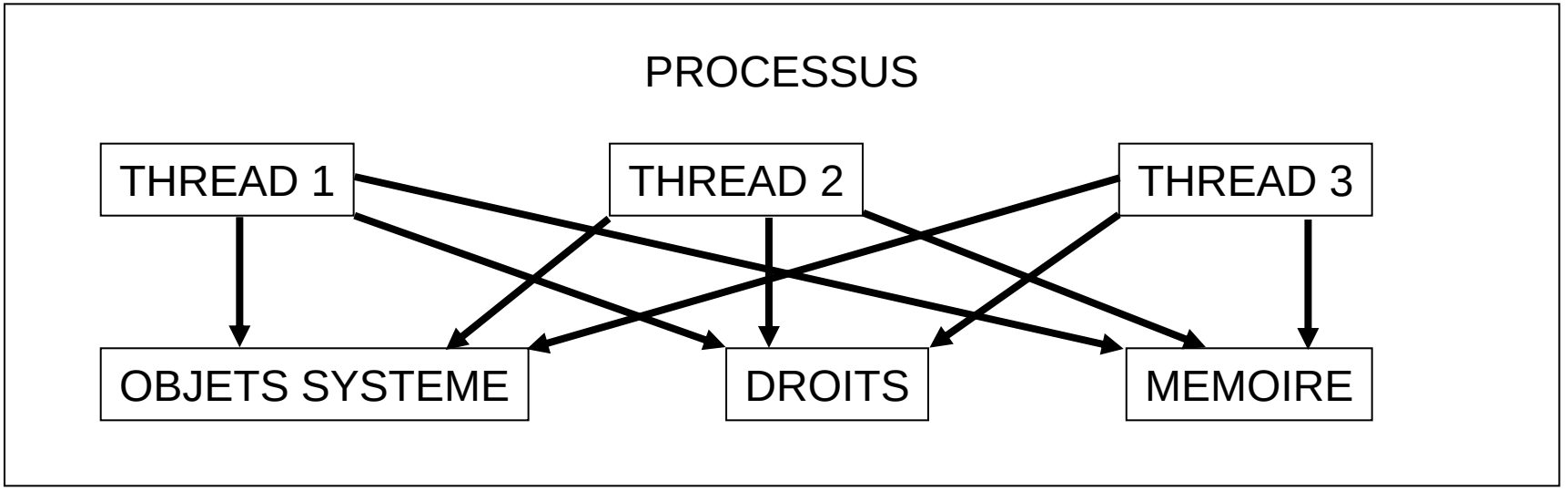
  - Sous Unix : fork
  - Sous Windows : CreateProcess

Les systèmes d'exploitation
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.

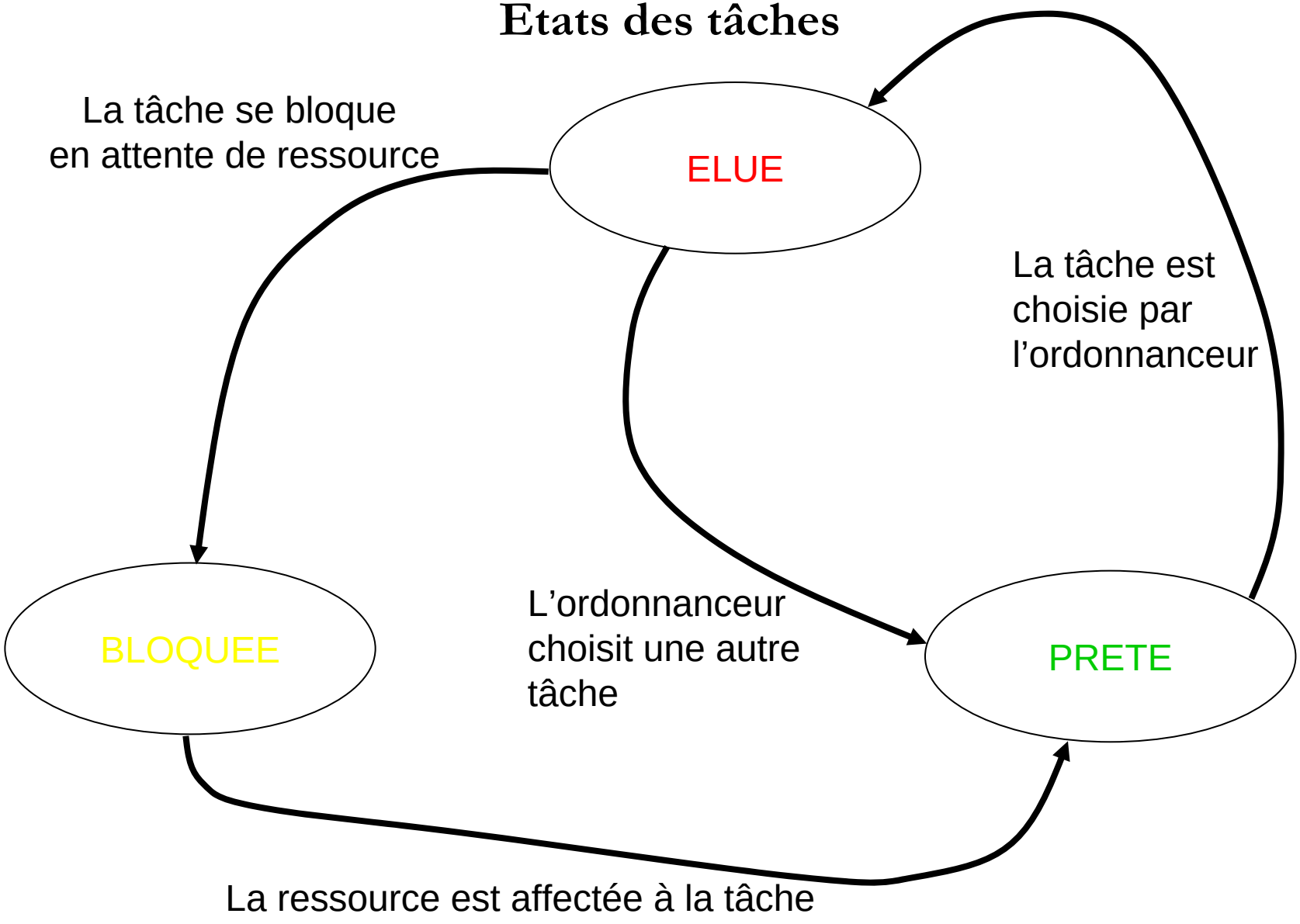
Les systèmes d'exploitation
Entités d'exécution

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.
- 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.

Les systèmes d'exploitation

Etats des tâches



Les systèmes d'exploitation
Communications inter – tâches

Les tâches communiquent entre elles :

- Par des mécanismes du système d'exploitation pour les processus :

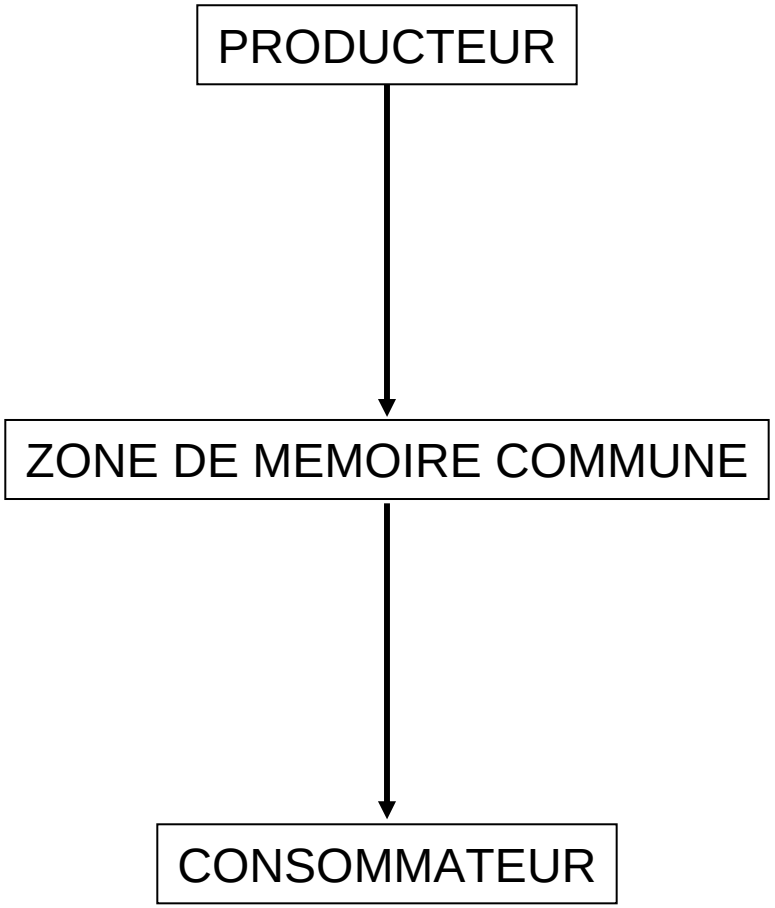
  => les tubes
  => les boites aux lettres
  => la mémoire partagée

- Par accès à des zones mémoires communes au sein du processus pour les threads :

  => Nécessité des mécanismes d'exclusions mutuelles

Les systèmes d'exploitation

Problème du producteur - consommateur



=> ACCES CONCURRENT A LA MEMOIRE COMMUNE

Les systèmes d'exploitation

Mécanisme de synchronisation des tâches

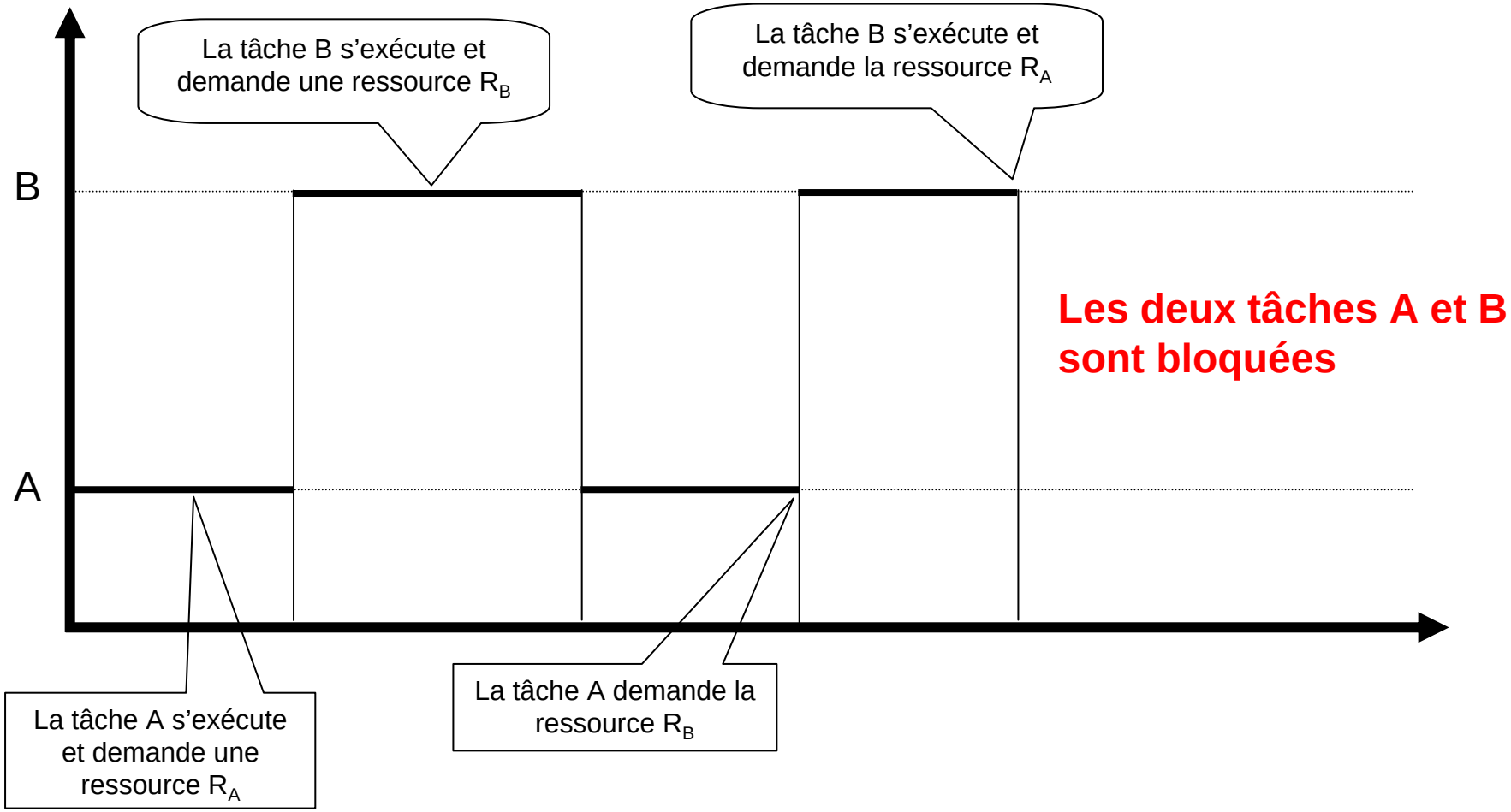
- Mutex
- Sémaphores
- Sections critiques
- Les rendez-vous (ADA)
- Les objets partagés

- Les boites aux lettres
- Les tubes

Les systèmes d'exploitation

Interblocage

Deux tâches peuvent se bloquer mutuellement :



Les systèmes d'exploitation

Ordonnancement

L'ordonnanceur doit :

⇒S'assurer que chaque tâche reçoit sa part du temps processeur.

⇒ Utiliser le temps processeur à 100%.

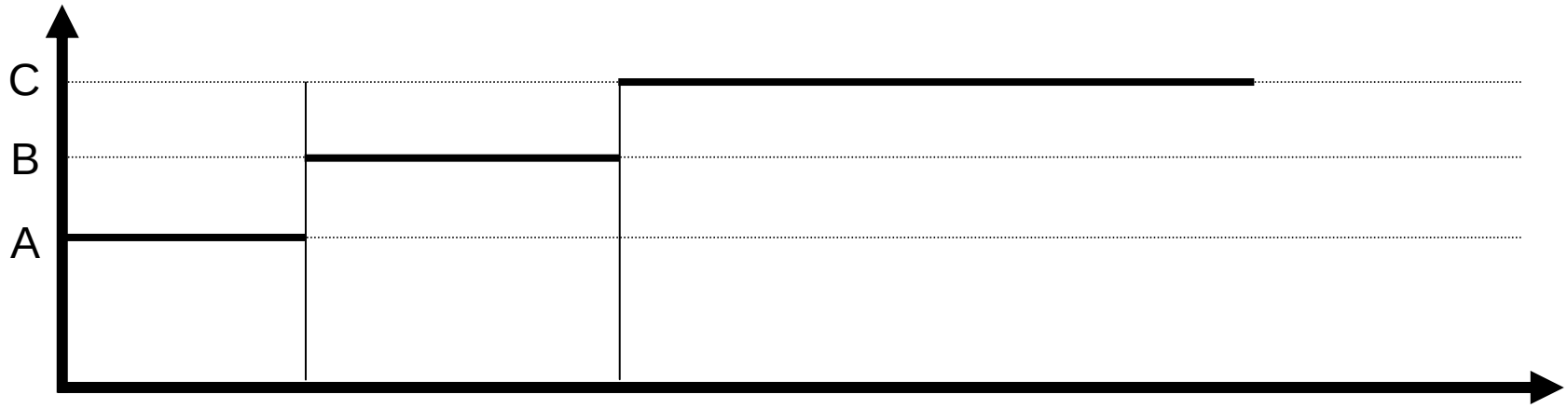
⇒ Minimiser les temps réponses pour les utilisateurs en mode interactif.

⇒ Maximiser le nombre de travaux effectués en une heure.

=> Il existe plusieurs politiques d'ordonnancement.

Les systèmes d'exploitation

Ordonnancement : premier arrivé – premier exécuté - FIFO

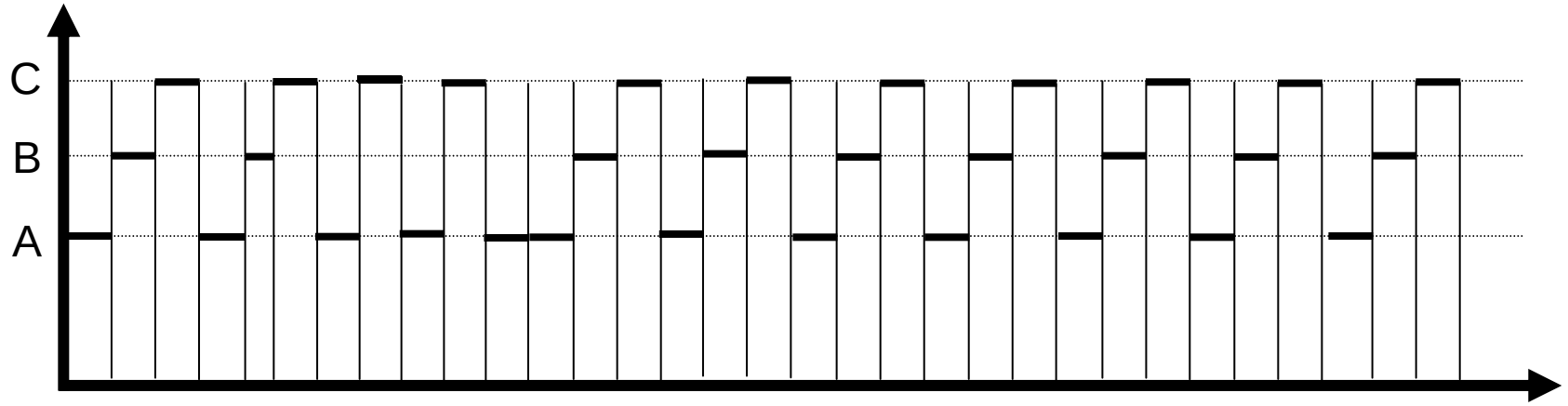


Les tâches sont exécutées les unes après les autres. Une tâche ayant la ressource processeur ne la rend que :

⇒ volontairement,
⇒ par blocage de la tâche suite à une demande d'entrées – sorties,
⇒ par terminaison de la tâche.

Les systèmes d'exploitation

Ordonnancement : tourniquet – Round Robin (RR)



⇒Temps partagé.

⇒ Le temps d'exécution est appelé **quantum de temps** ou **time slice**.

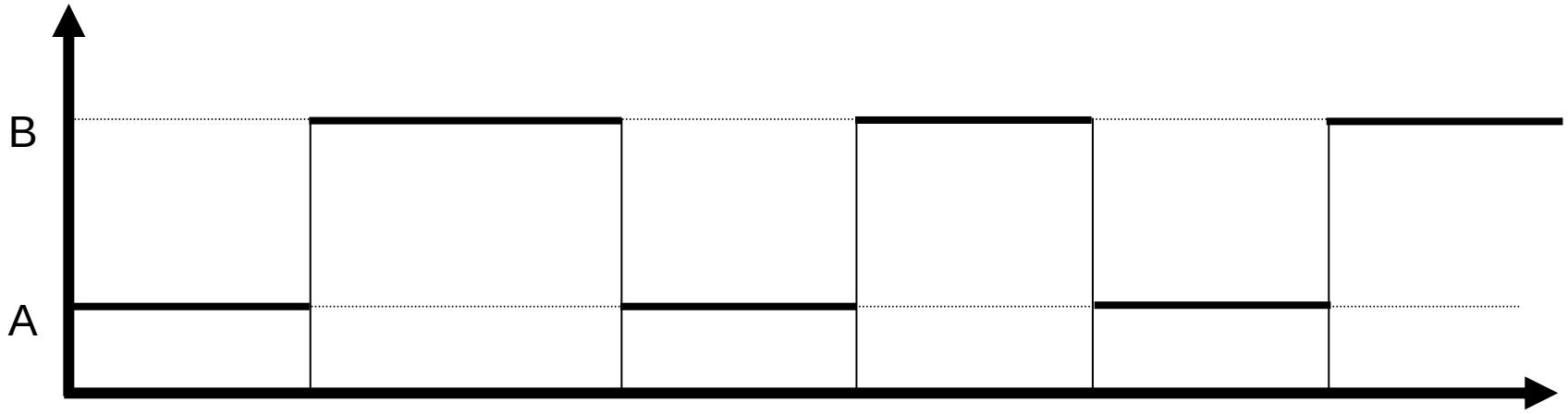
⇒ Le système utilise une liste et ordonnance les tâches suivants leurs positions dans cette liste.

⇒ 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.

Les systèmes d'exploitation

Ordonnancement par priorité



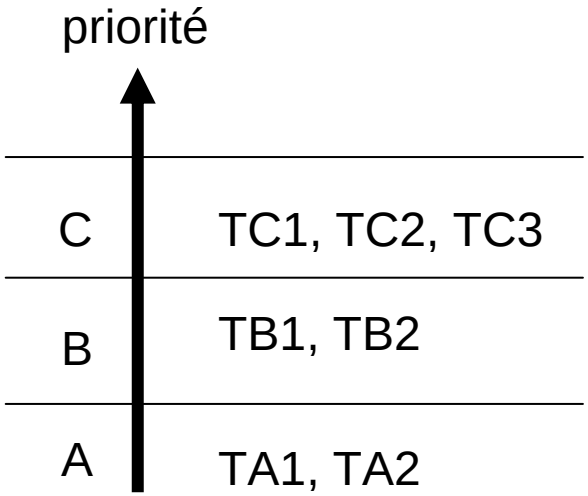
Chaque tâche dispose d'une valeur reflétant sa priorité. **Le système ordonnance la tâche de priorité ayant la plus haute priorité** (sous réserve qu'elle dispose de toutes les ressources).

Le système dispose d'une file par niveau de priorité. Dans cette file de niveau de priorité P sont mises les tâches de niveau de priorité P.

Les systèmes d'exploitation

Ordonnancement par priorité

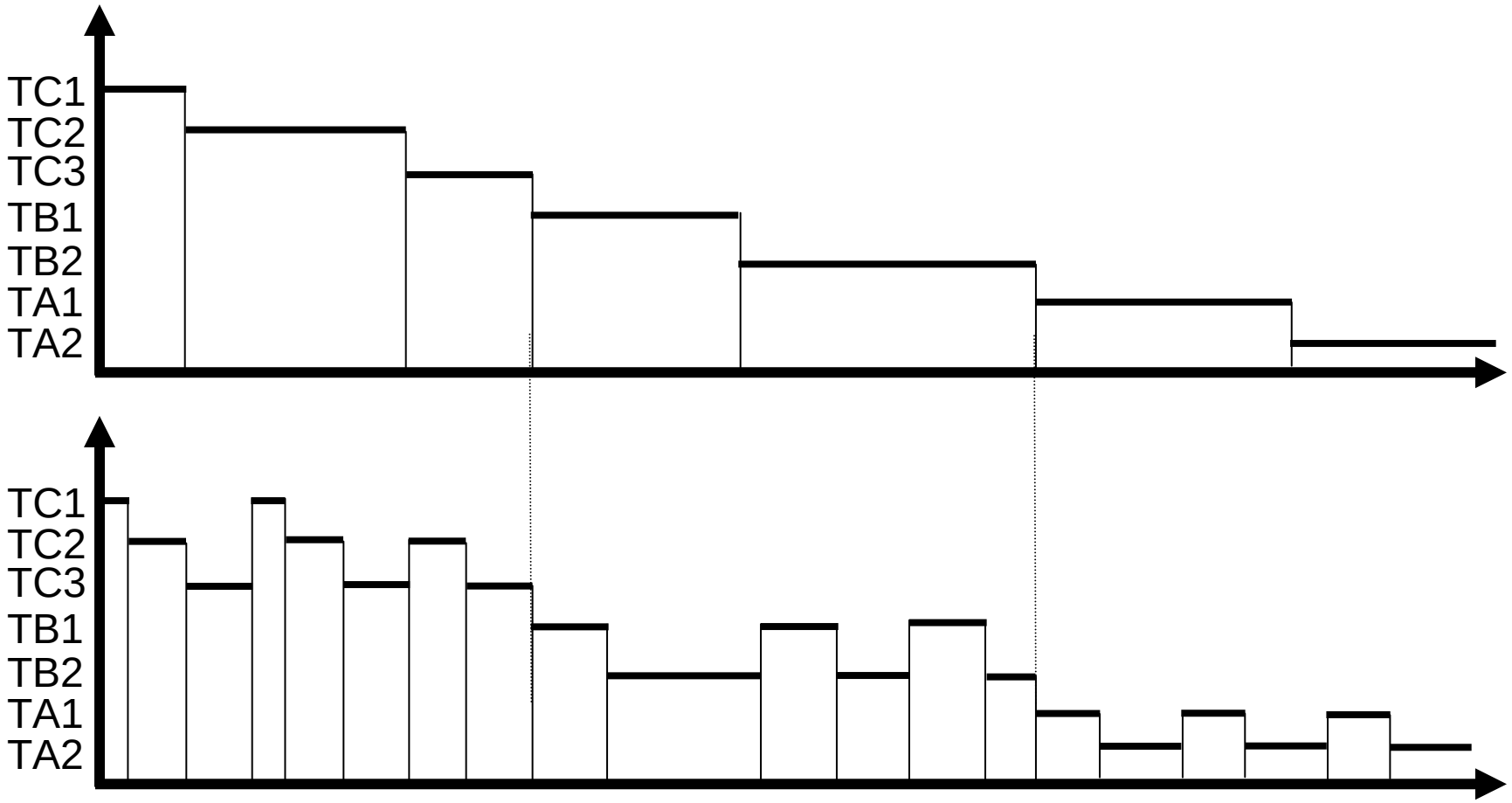
Si plusieurs tâches ont le même niveau de priorité et que ce niveau est le plus haut, l'ordonnancement de ces tâches peut être en FIFO ou en ROUND ROBIN.



Trouvez l'ordonnancement des tâches ci-dessus en tourniquet puis en fifo.

Les systèmes d'exploitation

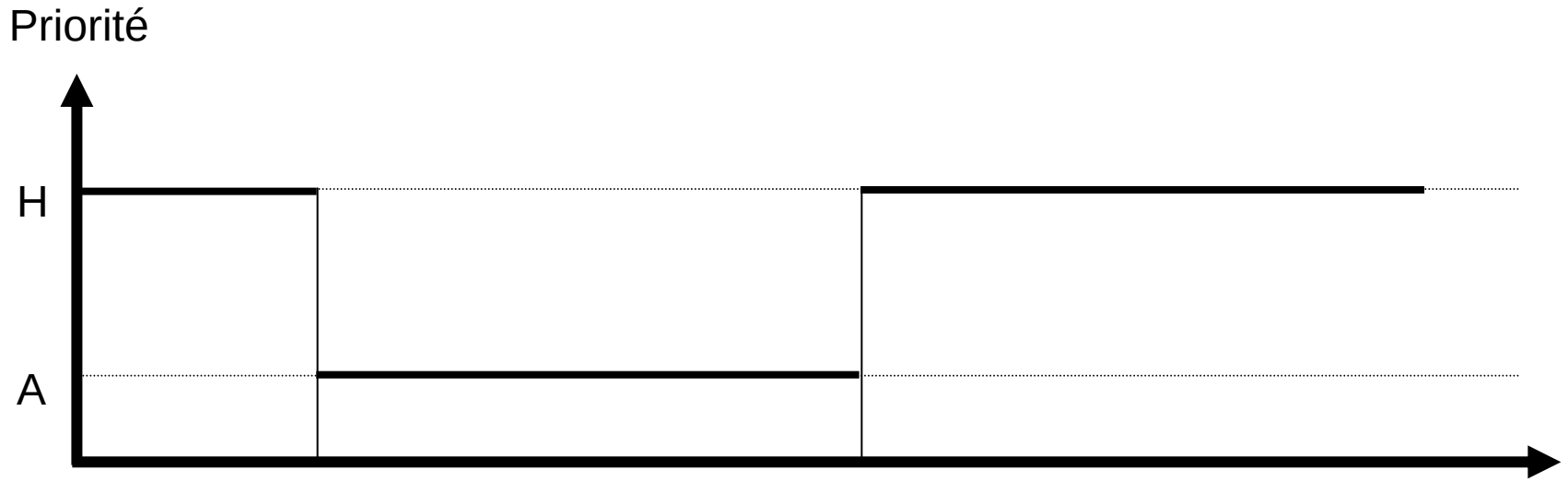
Ordonnancement par priorité



Les systèmes d'exploitation

Ordonnancement 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** pré-emptant toutes les tâches de priorité inférieur.



Les systèmes d'exploitation

Ordonnancement coopératif

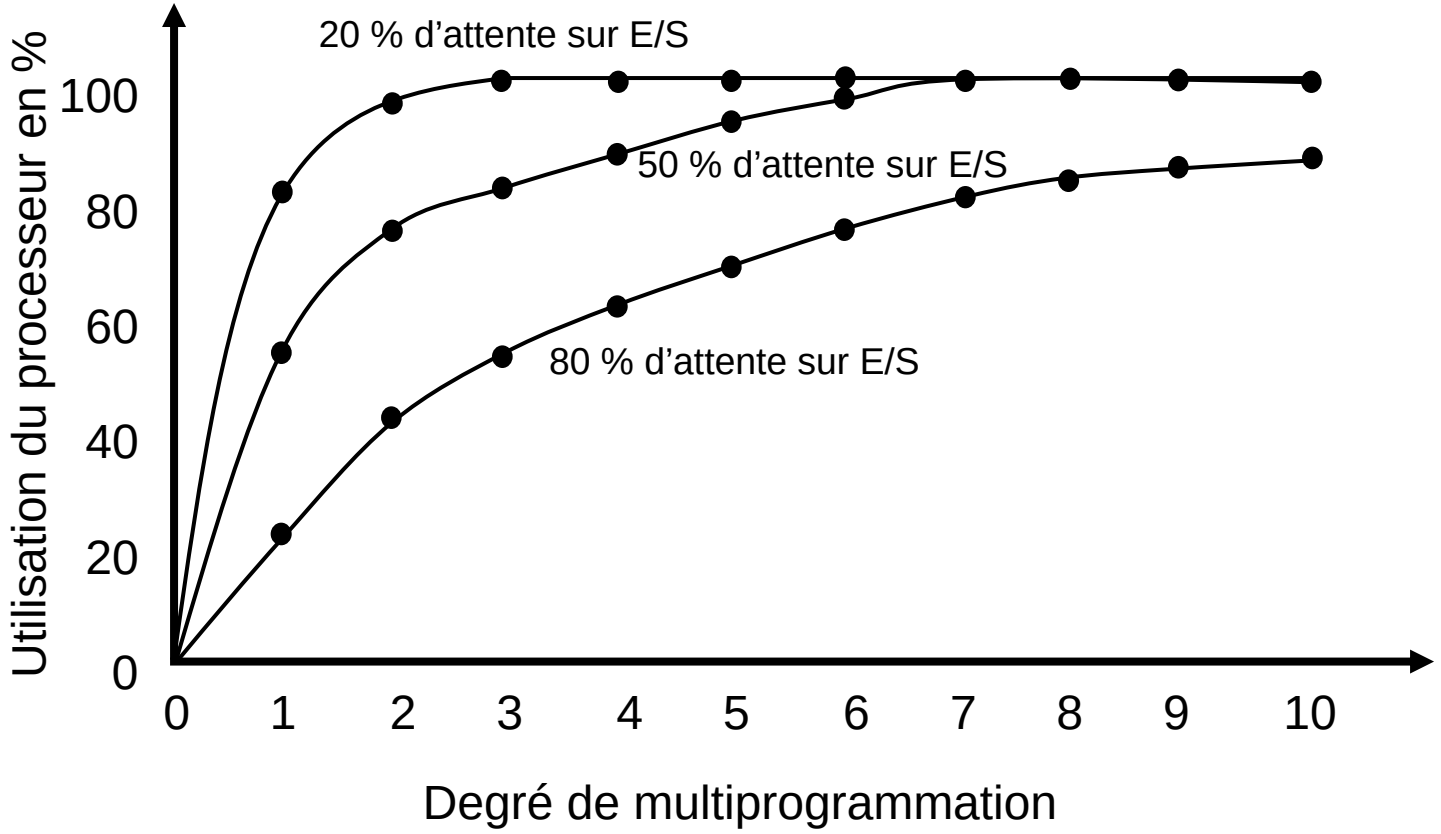
Dans ce type de système, ce sont les tâches qui décident de libérer volontairement le processeur (sauf si elles sont bloquées par une E/S).

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.

Les systèmes d'exploitation

Utilisation du temps UC



Les systèmes d'exploitation

- 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.

Problématique :

Système multi – utilisateur :

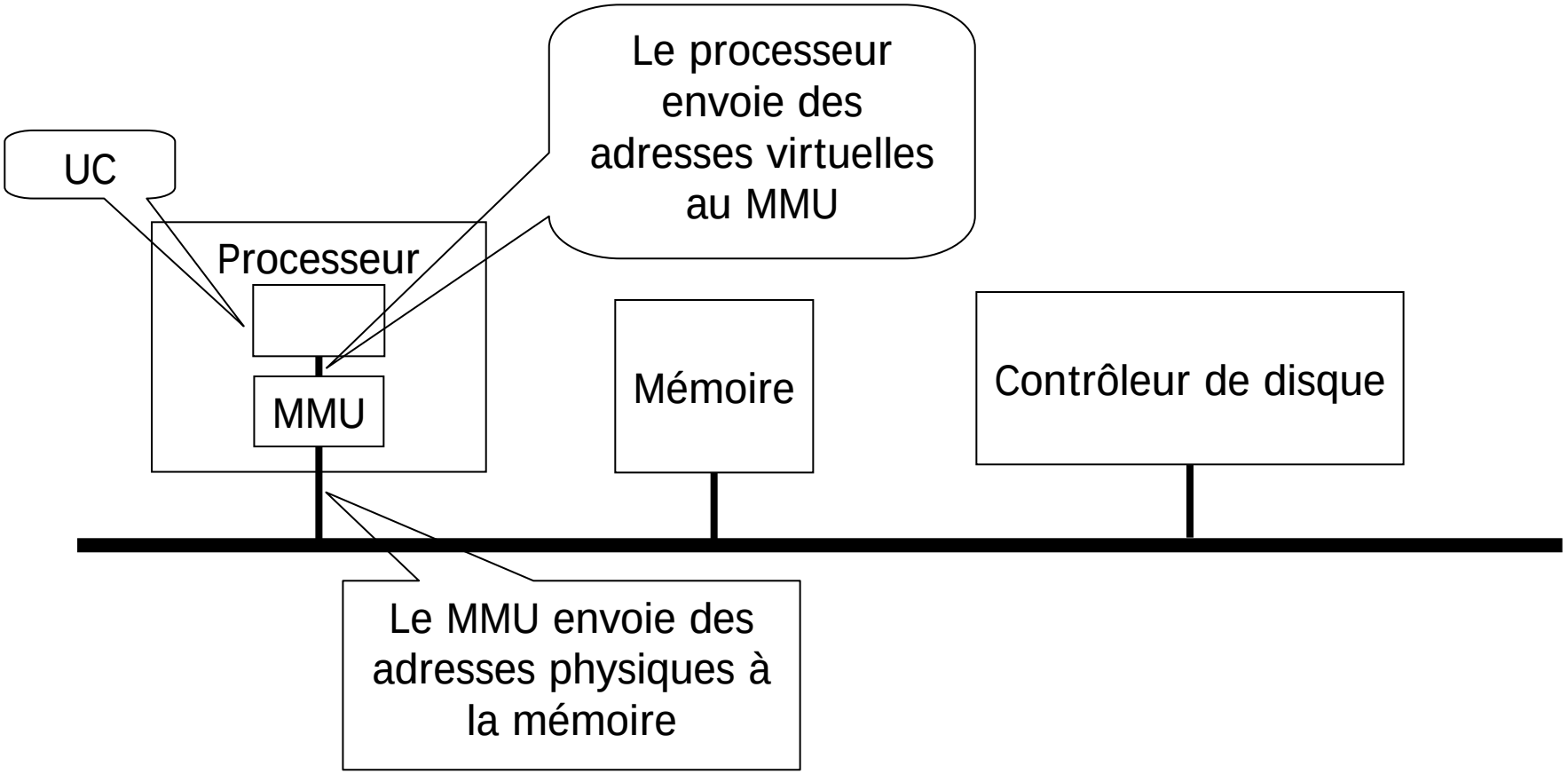
⇒ Système multi – tâche.
⇒ De grande capacité mémoire dépassant la mémoire physique de la machine.

Solution : la mémoire virtuelle

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

Les systèmes d'exploitation

La mémoire virtuelle



Les systèmes d'exploitation

La pagination

L'espace d'adressage virtuelle :

- Les adresses manipulés par les processus.
- L'ensemble des adresses virtuelles.

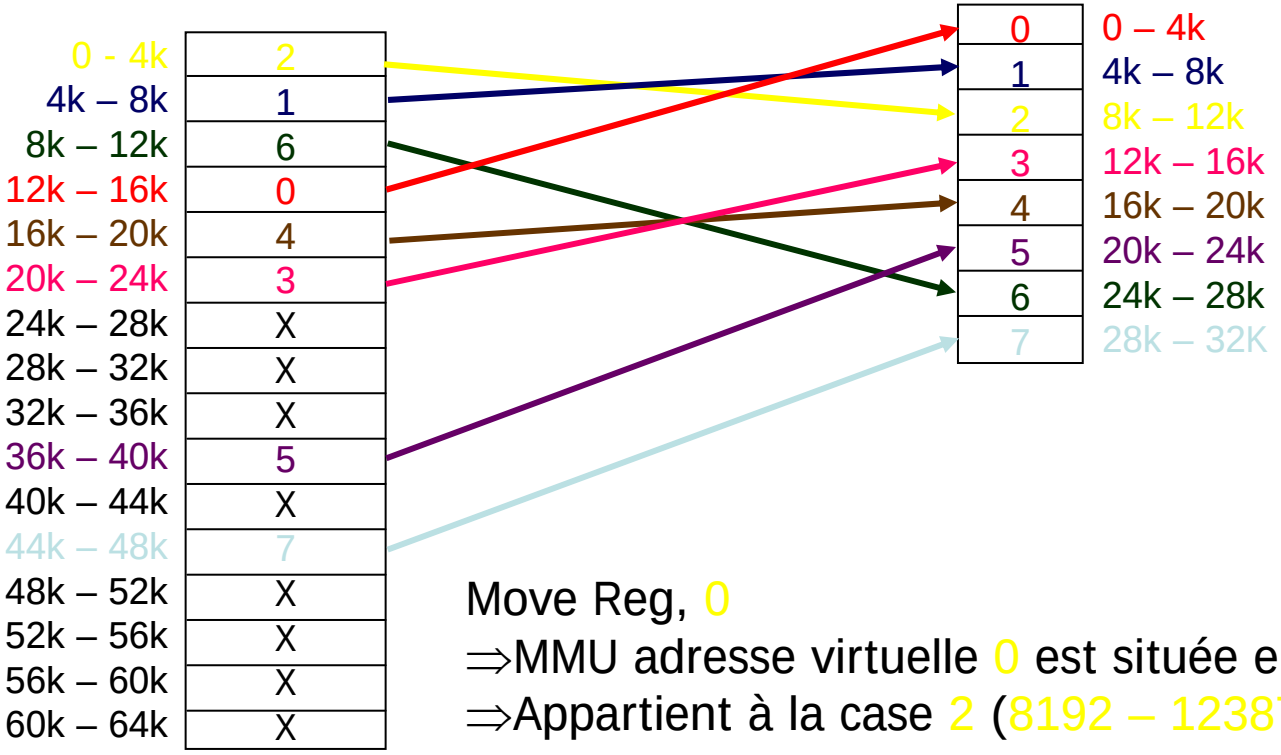
La mémoire virtuelle utilise la technique de la pagination :

Cette technique consiste a divisé l'espace d'adressage virtuelle 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).

Les systèmes d'exploitation

La pagination



Move Reg, 0

⇒MMU adresse virtuelle 0 est située en page 0 (0-4095)

⇒Appartient à la case 2 (8192 – 12387)

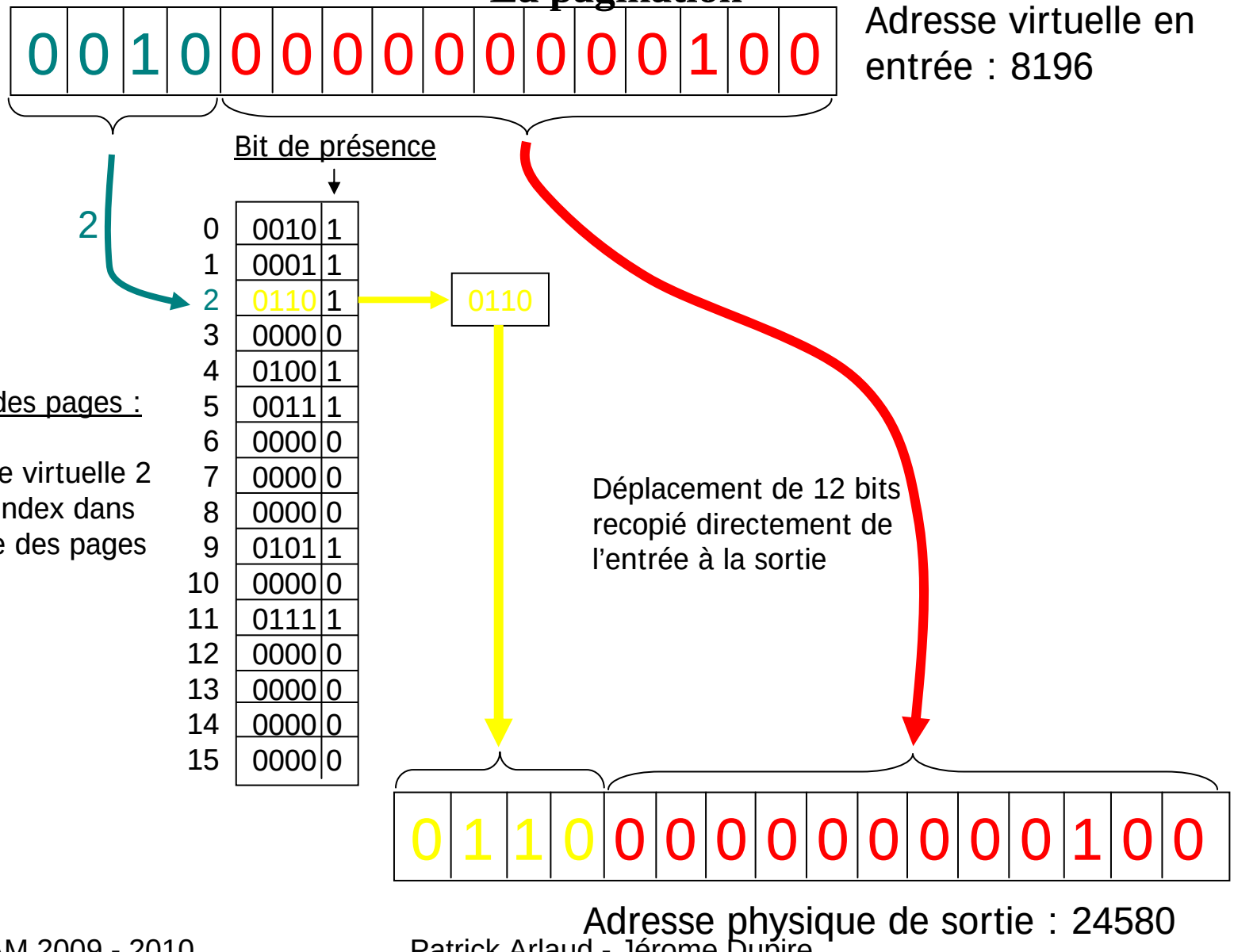
Le MMU transforme l'adresse 0 en 8192 : MAPPAGE

Move Reg, 8192 => Move Reg, 24576

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

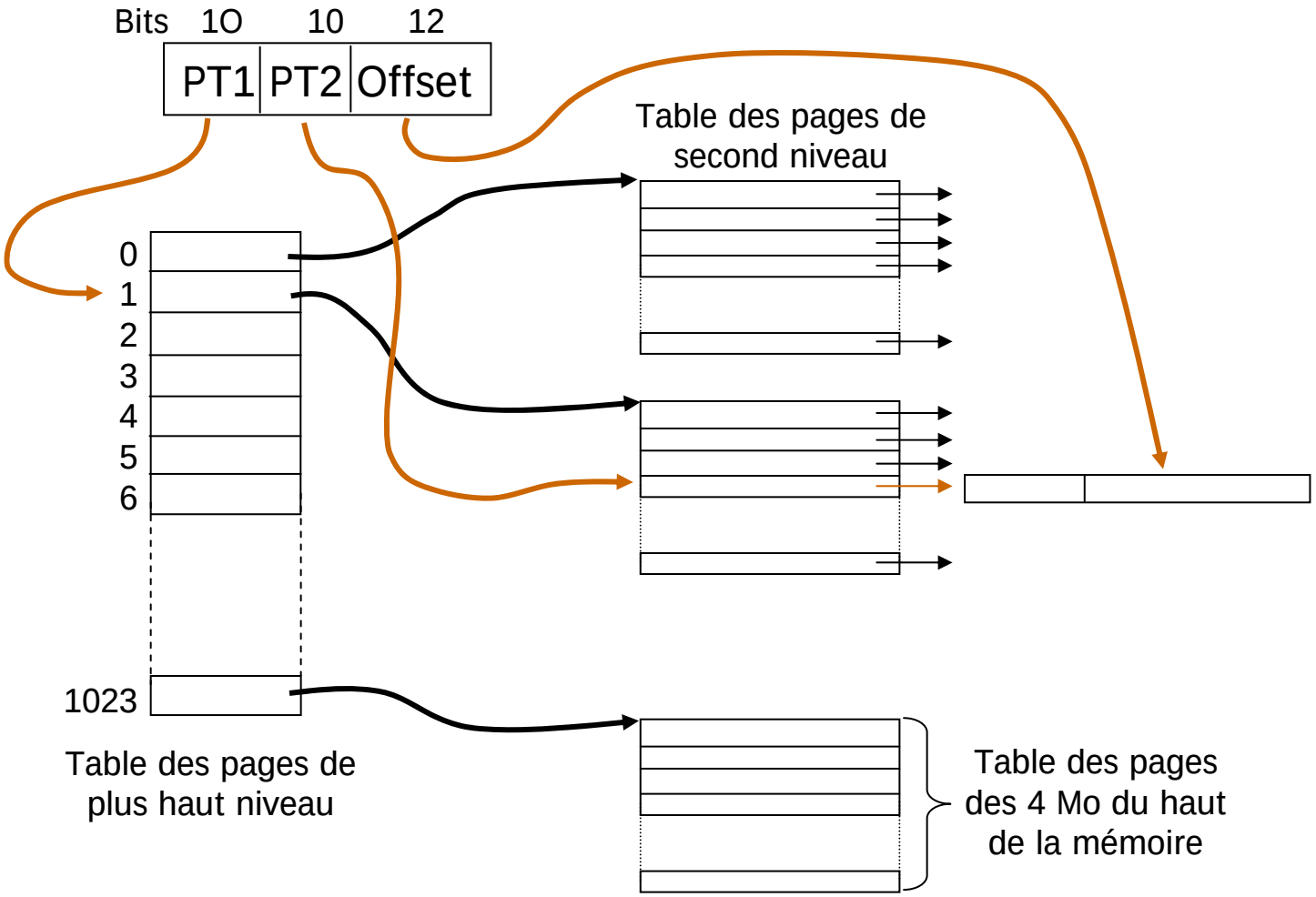
Les systèmes d'exploitation

La pagination



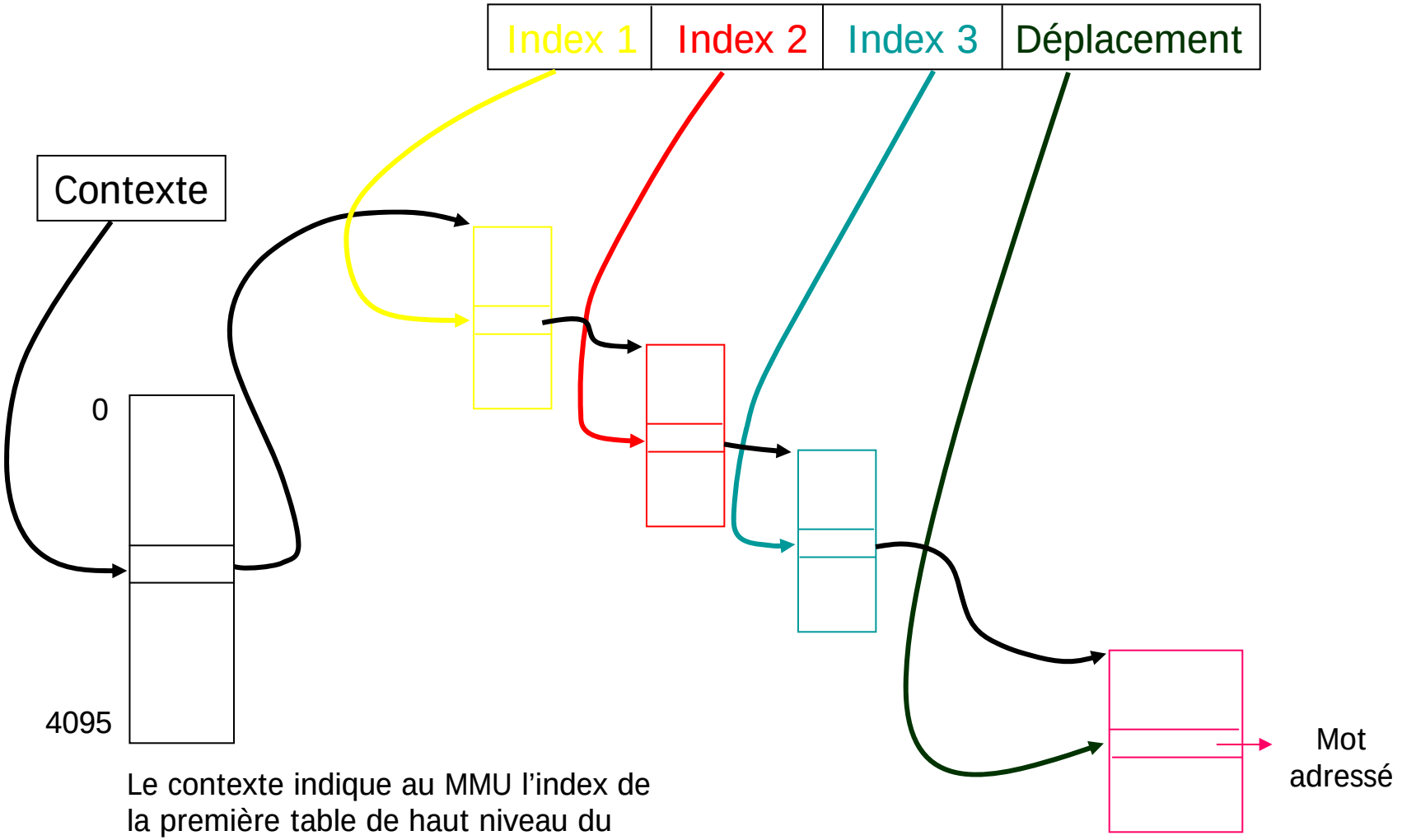
Les systèmes d'exploitation

La mémoire virtuelle à plusieurs niveaux



Les systèmes d'exploitation

Exemple : le Sparc

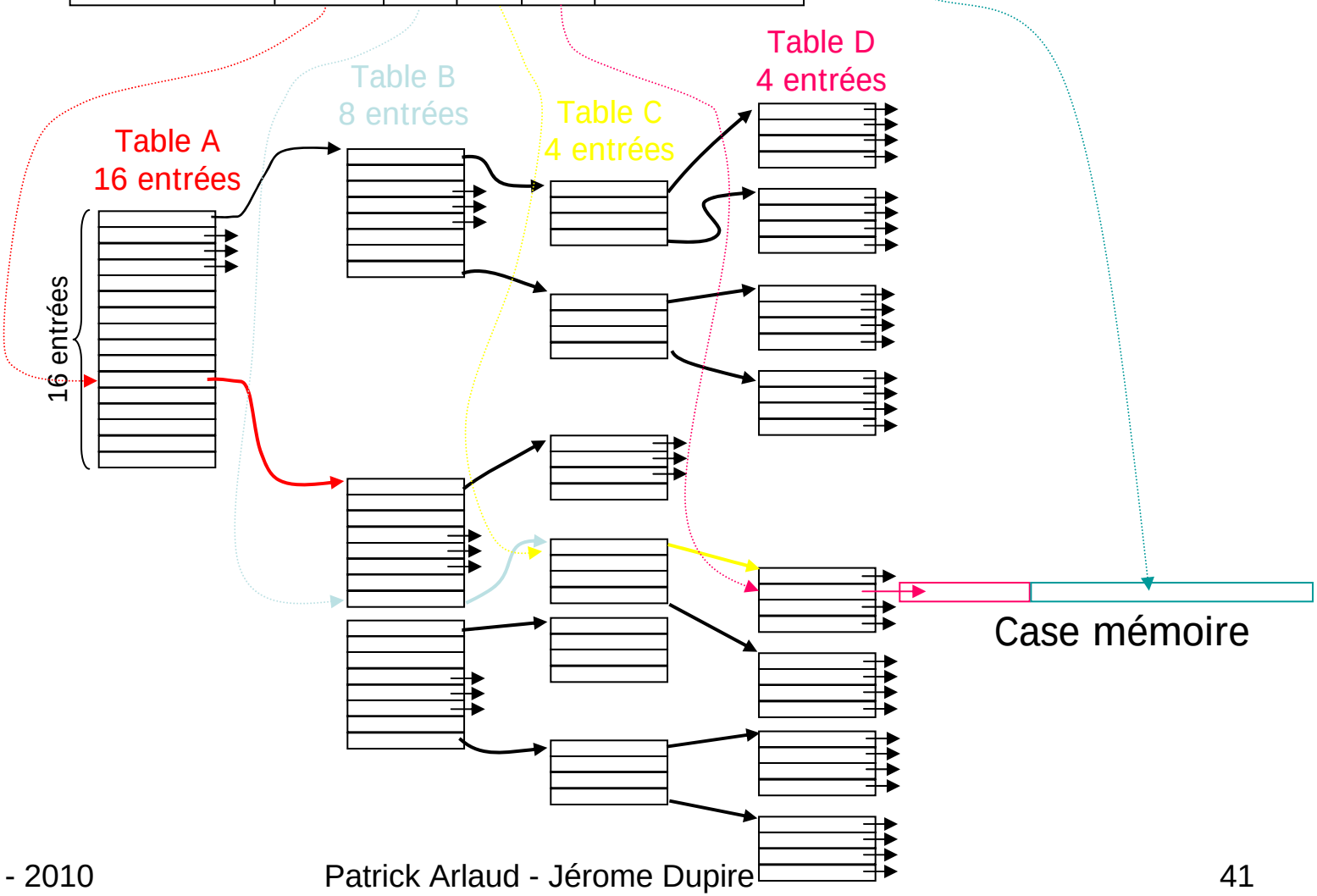


Le contexte indique au MMU l'index de la première table de haut niveau du processus en cours d'exécution.

Les systèmes d'exploitation

Exemple : le 68030

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



Les systèmes d'exploitation

Mémoire associative

C'est une mémoire 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.

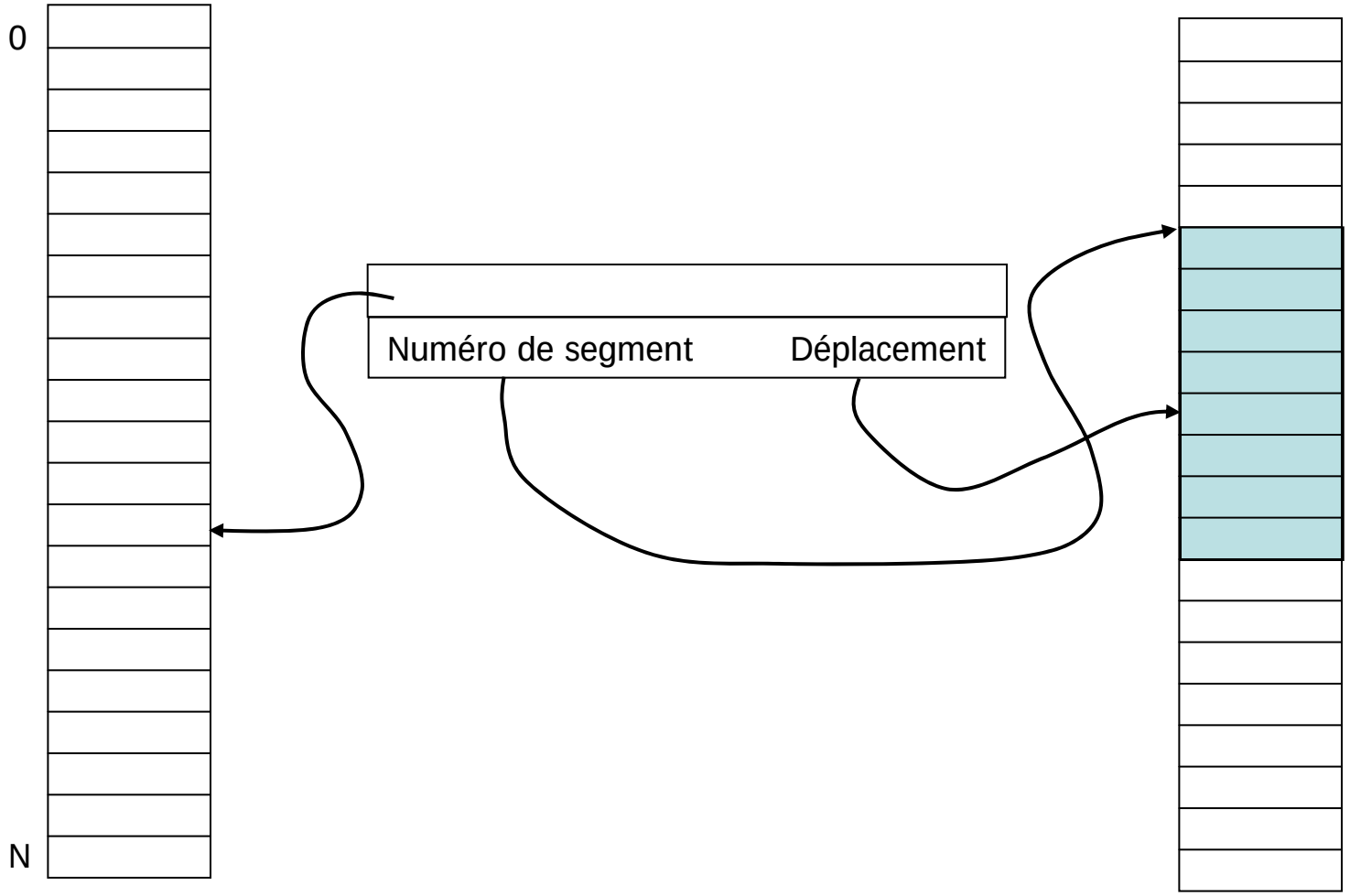
Indicateur de validité
Pages virtuelles
Indicateur de modification
Indicateur de protection
Case mémoire

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

Les systèmes d'exploitation

La segmentation

La mémoire apparaît comme un espace à deux dimensions



Les systèmes d'exploitation

Exemple de segmentation : Intel 8086

Des registres de segment de 16 bits
Des registres de déplacement de 16 bits

Adresse logique dans deux registres de 16 bits :

```
          0000 0000 0000 0000        Registre de segment
+         0000 0000 0000 0000        Déplacement

          0000 0000 0000 0000 0000
```

Pour adresse la case mémoire de l'adresse physique 0x10004 (65540) :

⇒ Registre de segment : 0001 0000 0000 0000
⇒ Registre de déplacement : 0000 0000 0000 0100

Les systèmes d'exploitation

Segmentation

La segmentation a été créée 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 systèmes d'exploitation

Pagination - segmentation

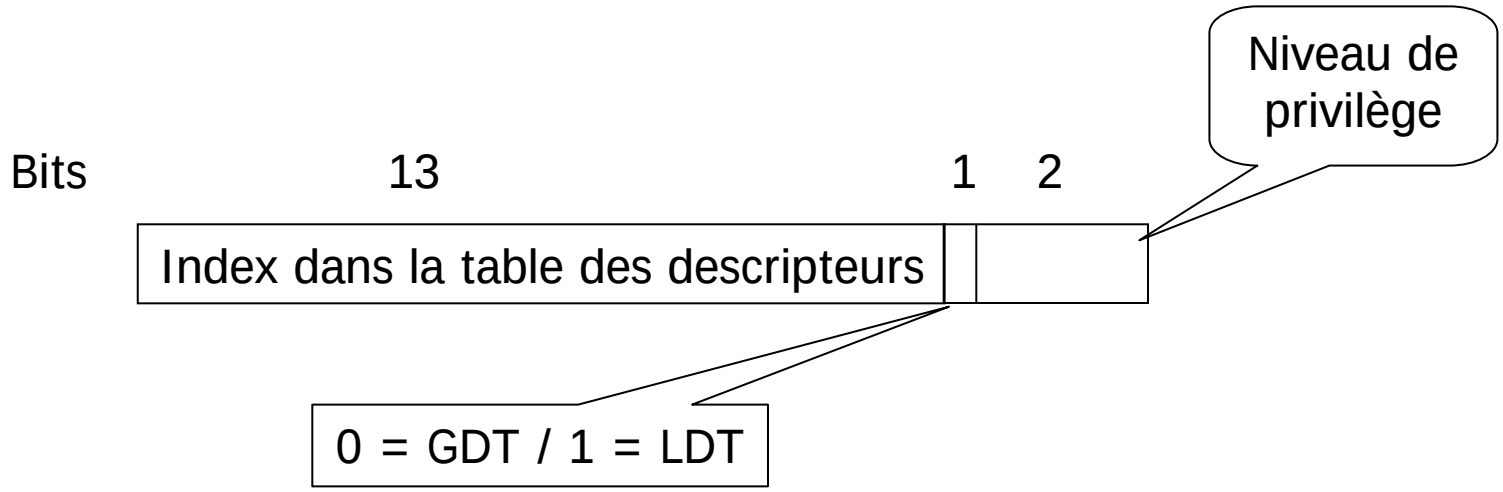
Considérations	Pagination	Segmentation
Le programmeur doit-il connaître la technique utilisée ?	Non	Oui
Combien y a-t-il d'espaces d'adressage linéaire ?	1	Plusieurs
L'espace d'adressage peut-il dépasser la taille de la mémoire physique ?	Non	Oui
Les procédures et les données peuvent elles être distinguées 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 sans avoir à acheter de la mémoire physique	Pour permettre la séparation des programmes et des données dans des espaces d'adressage indépendants et pour faciliter le partage et la protection

Les systèmes d'exploitation

Exemple de segmentation avec pagination : 80386

La table des descripteurs locaux : LDT (Local Descriptor Table) est unique pour chaque programme. La table des descripteurs globaux GDT (Global Descriptor Table) est commune au système.

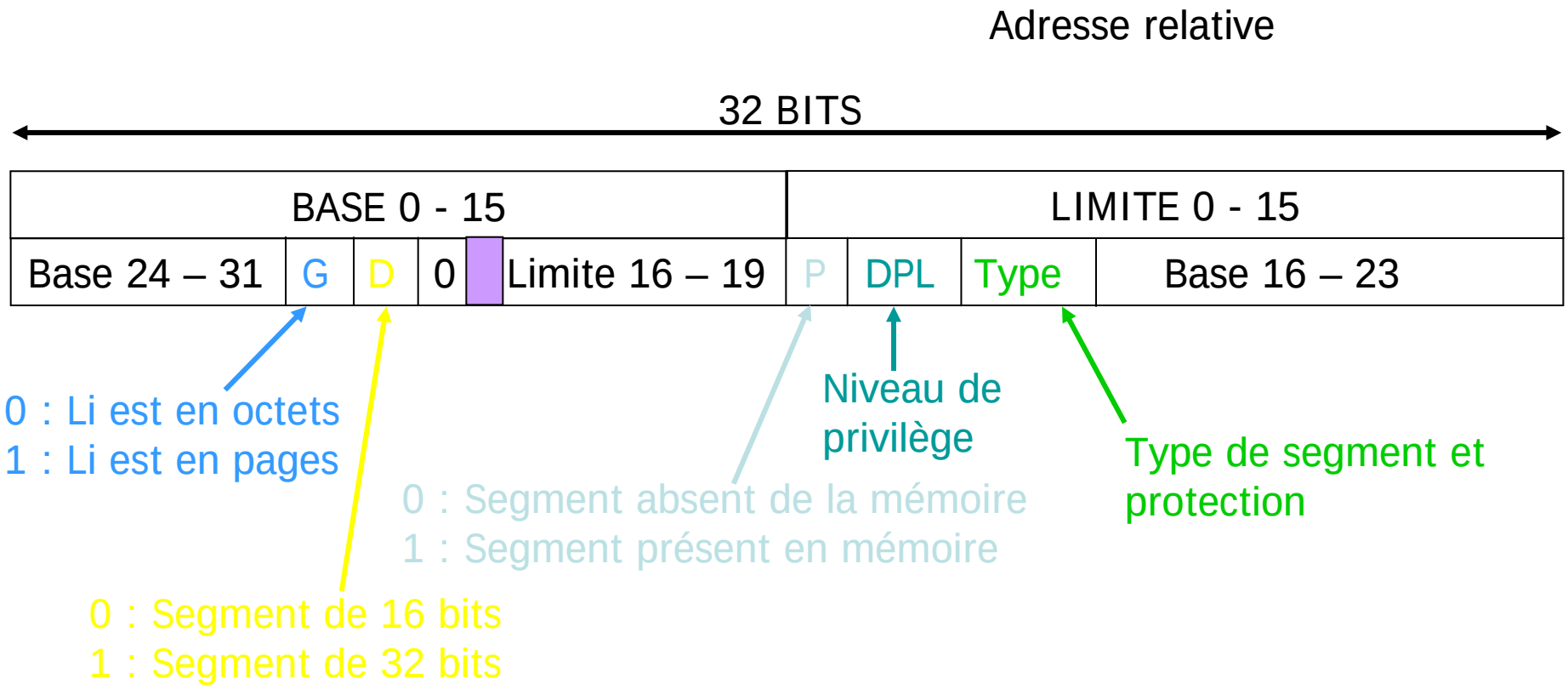
Le 80386 accède à un segment à travers son sélecteur en le chargeant dans un registre de segment :



Les systèmes d'exploitation

Exemple de segmentation avec pagination : 80386

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

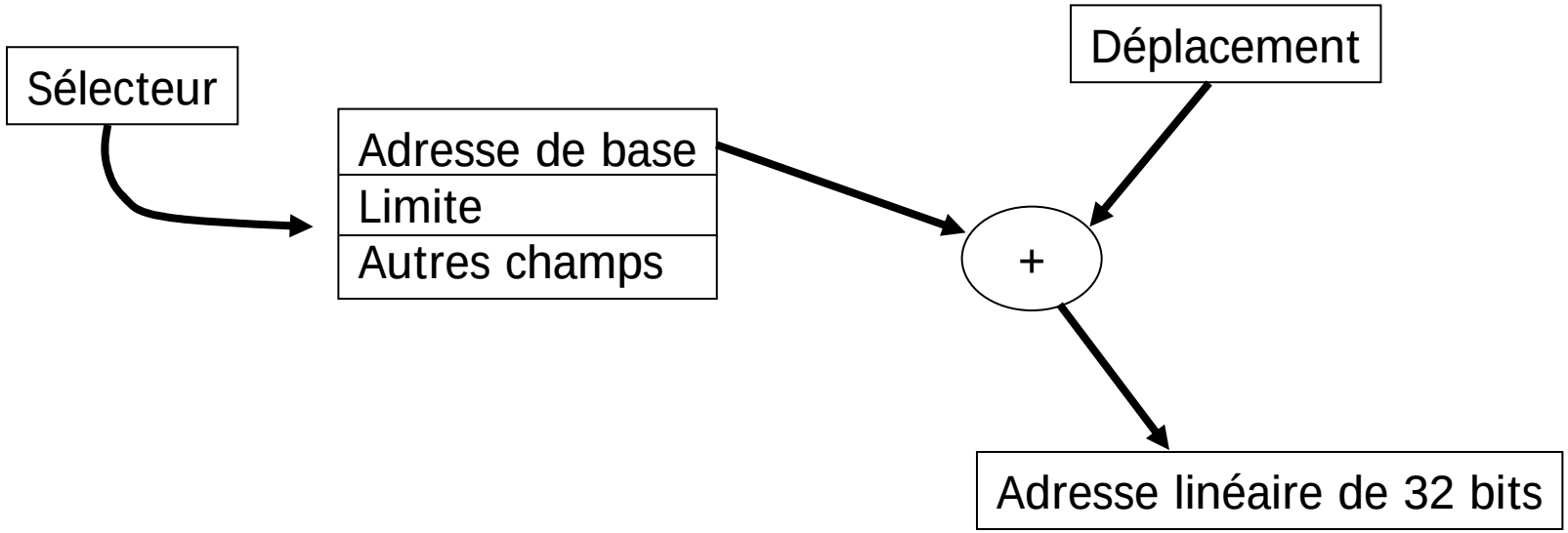


Les systèmes d'exploitation

Exemple de segmentation avec pagination : 80386

(Sélecteur, déplacement) :

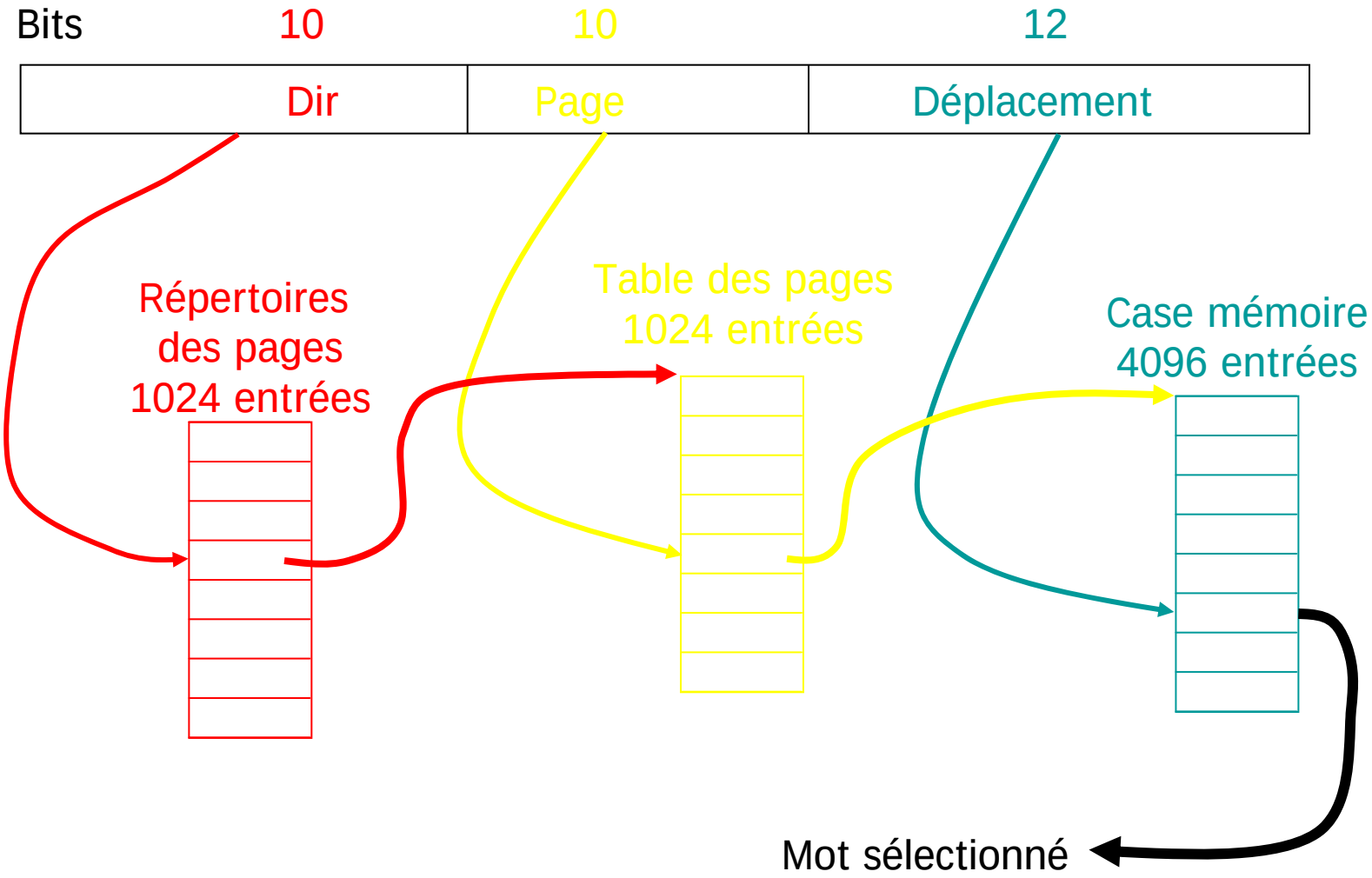
- Trouver le descripteur du segment répertorié par le sélecteur
- Vérifier si le déplacement dépasse la fin du segment (champ limite)
- Ajouter les 32 bits de base au déplacement de façon à former l'adresse linéaire



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

Les systèmes d'exploitation

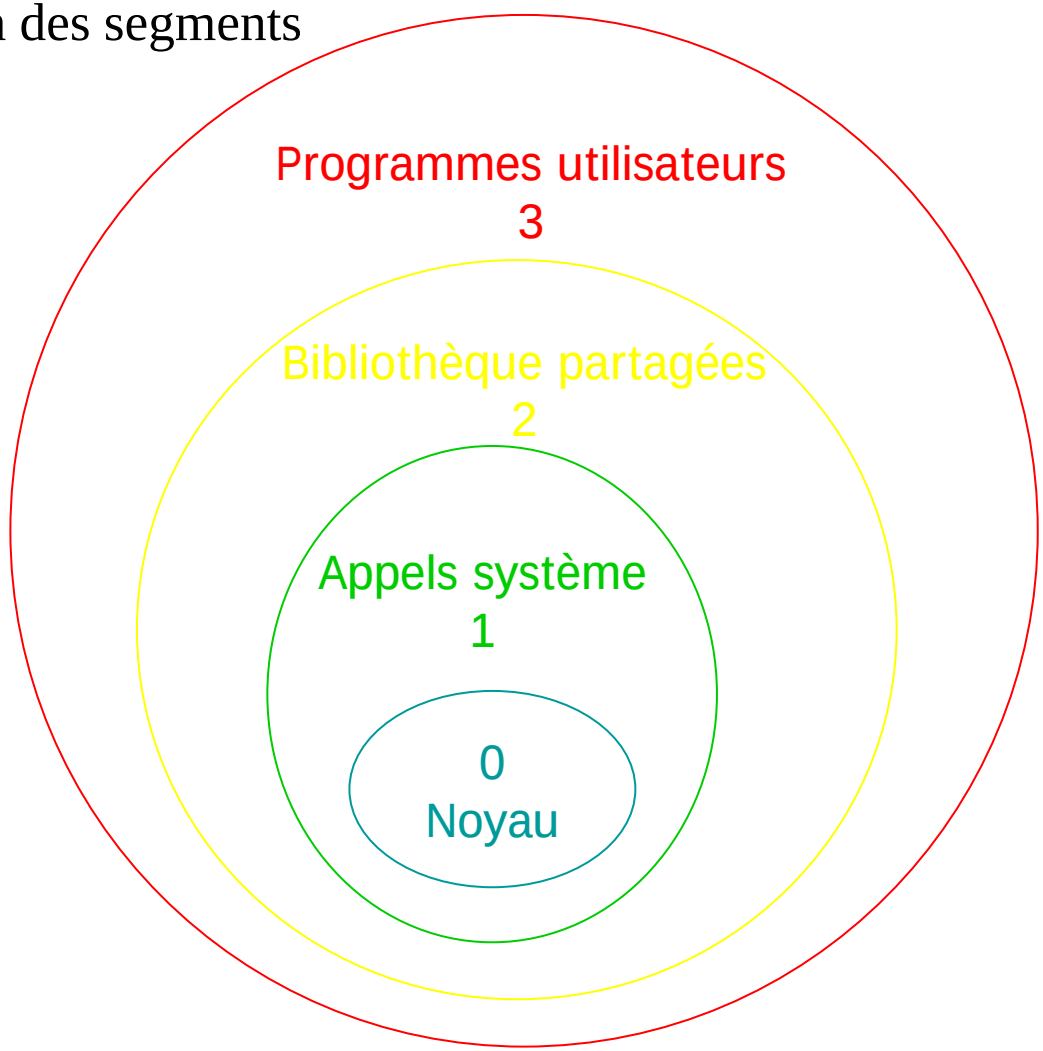
Exemple de segmentation avec pagination : 80386



Les systèmes d'exploitation

Exemple de segmentation avec pagination : 80386

Exemple de protection des segments



Les systèmes d'exploitation

Algorithme NRU

Algorithme de remplacement d'une page non récemment utilisée

Régulièrement 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

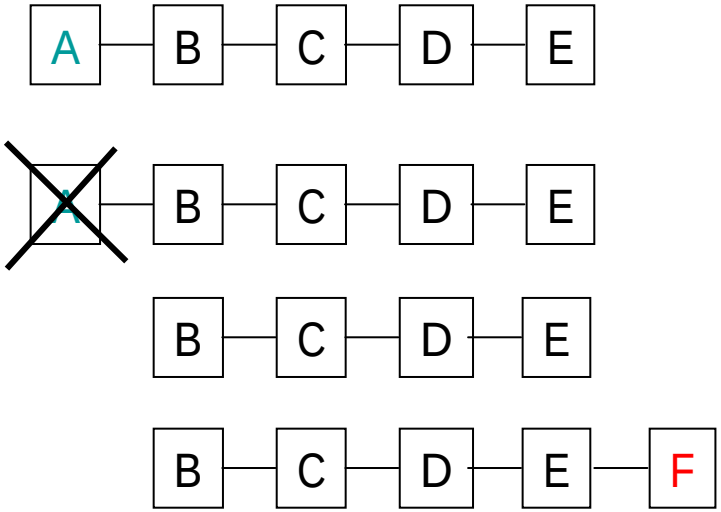
L'algorithme (NRU Not Recently Used) retire une page au hasard de la catégorie non vide qui a le plus petit numéro

Les systèmes d'exploitation

Algorithme 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.

Algorithme LRU

Une page ayant servie récemment servira certainement prochainement.

Une page n'ayant pas servie récemment ne servira certainement pas prochainement.

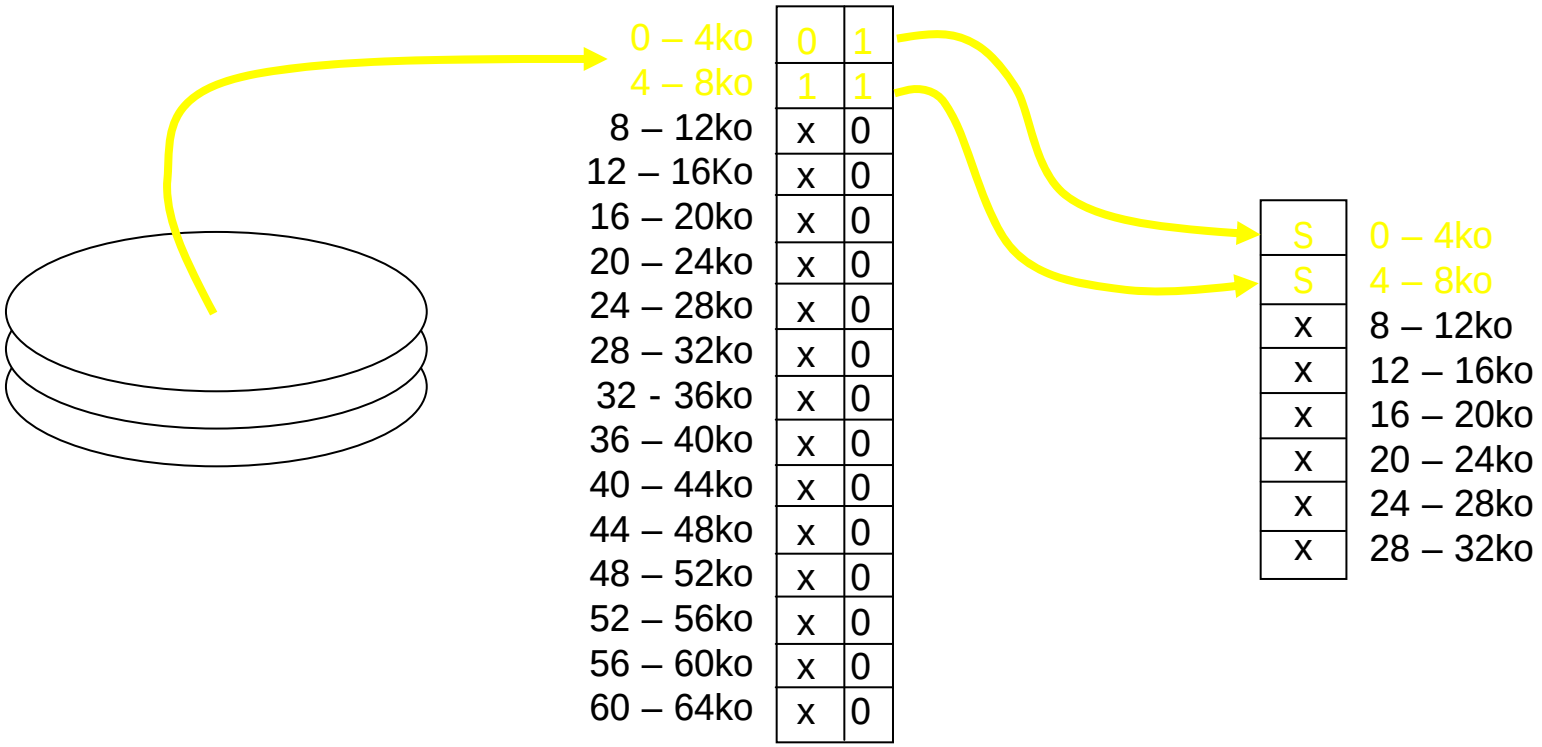
⇒Suppression de la page qui est restée inutilisée pendant le plus de temps.

LRU : Least Recently Used

Les systèmes d'exploitation

Exemple

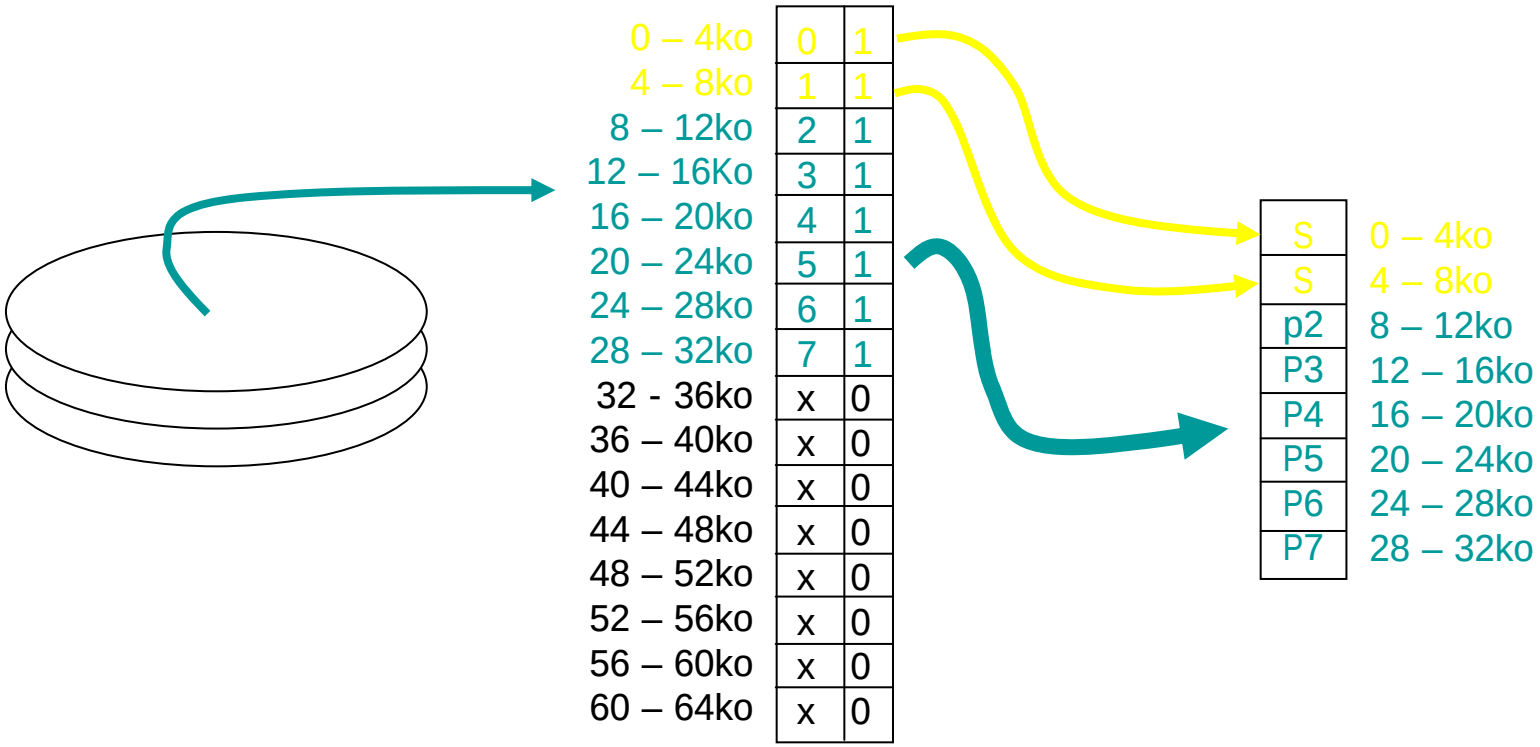
Au démarrage du système



Les systèmes d'exploitation

Exemple

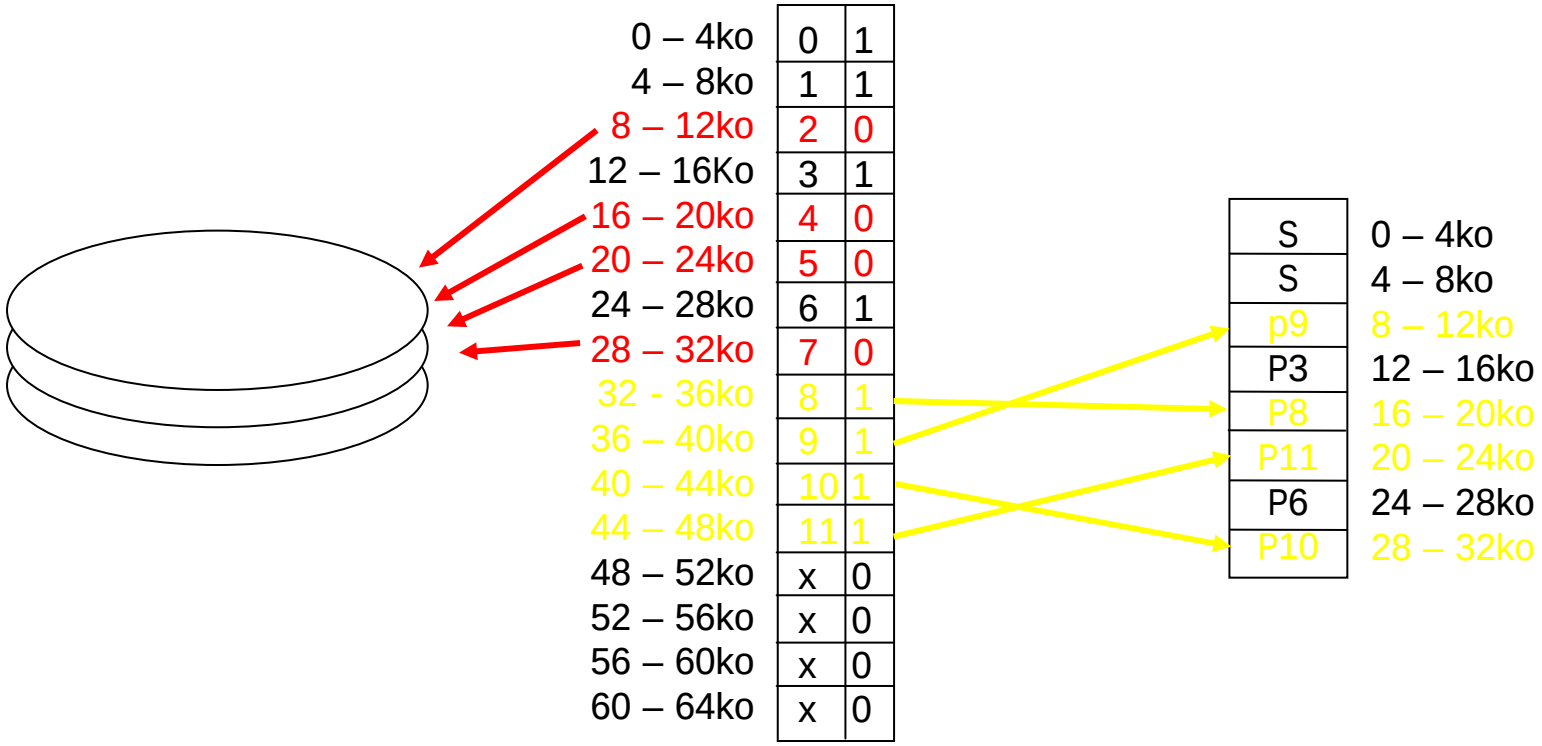
Après lancement d'un processus consommant de la mémoire



Les systèmes d'exploitation

Exemple

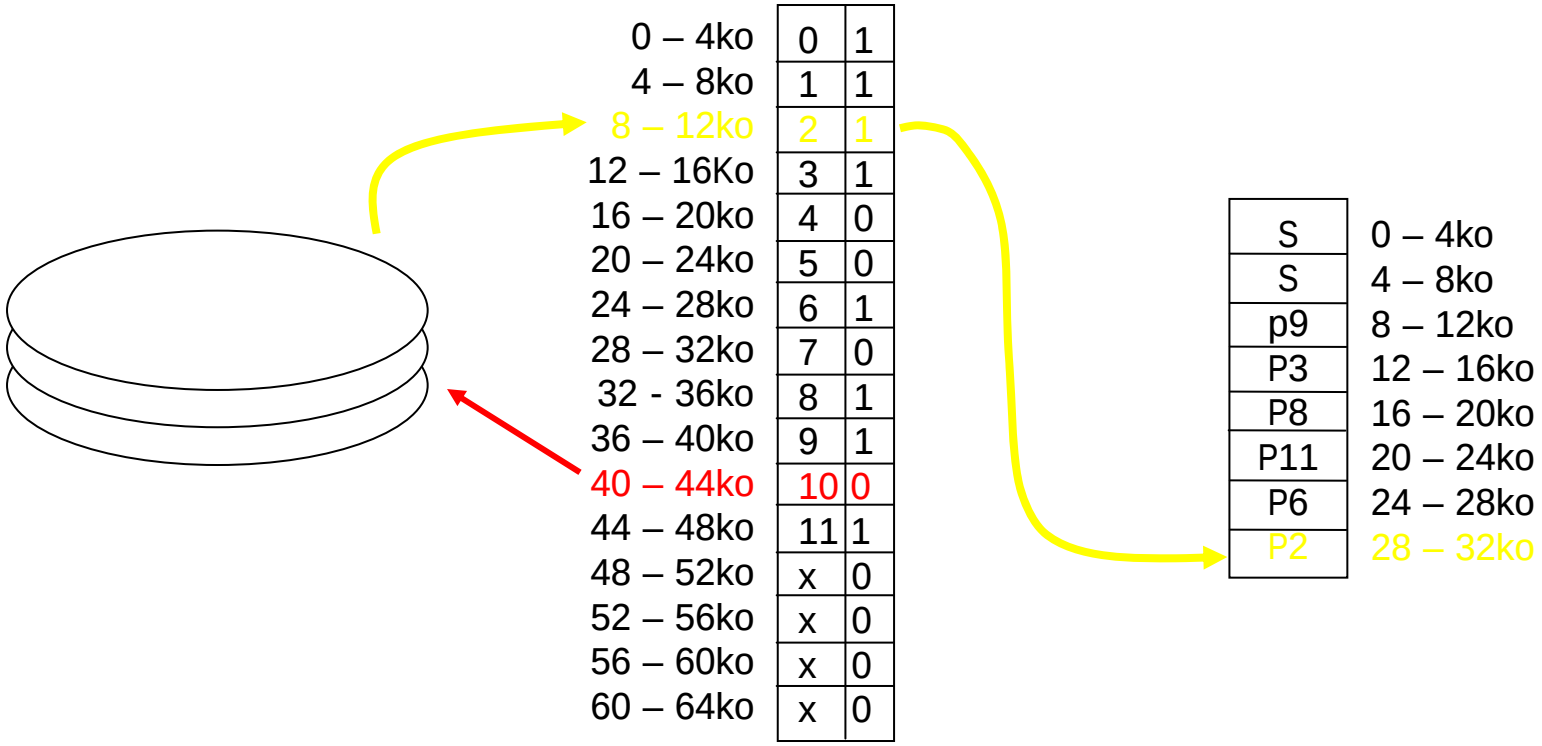
Après quelques secondes d'exécution, le processus demande 16ko de mémoire



Les systèmes d'exploitation

Exemple

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

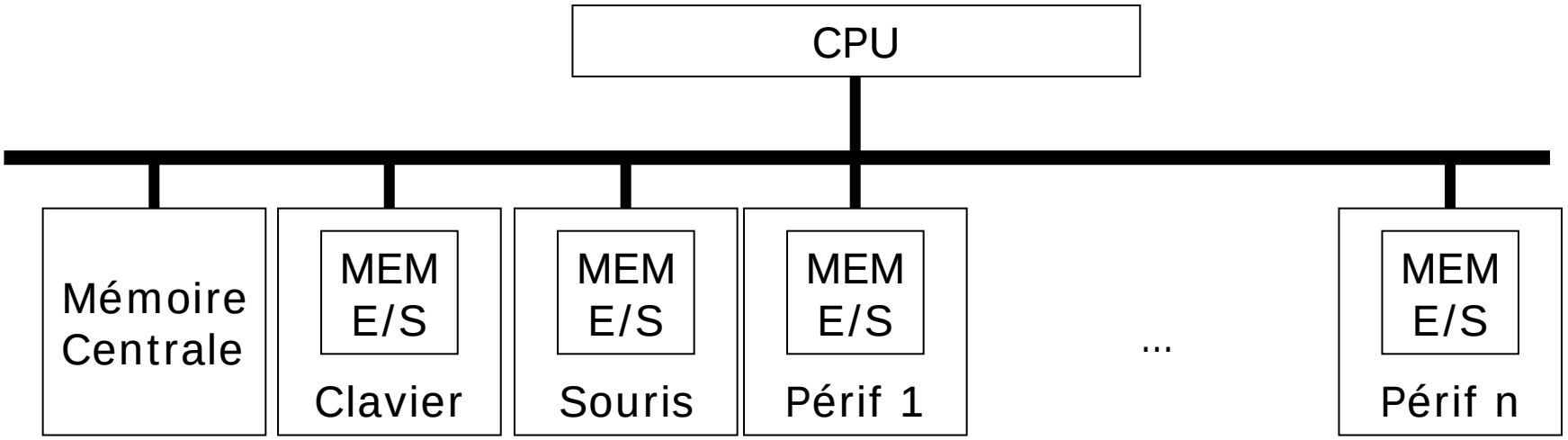


Les systèmes d'exploitation

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.



Les systèmes d'exploitation

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.

Les systèmes d'exploitation

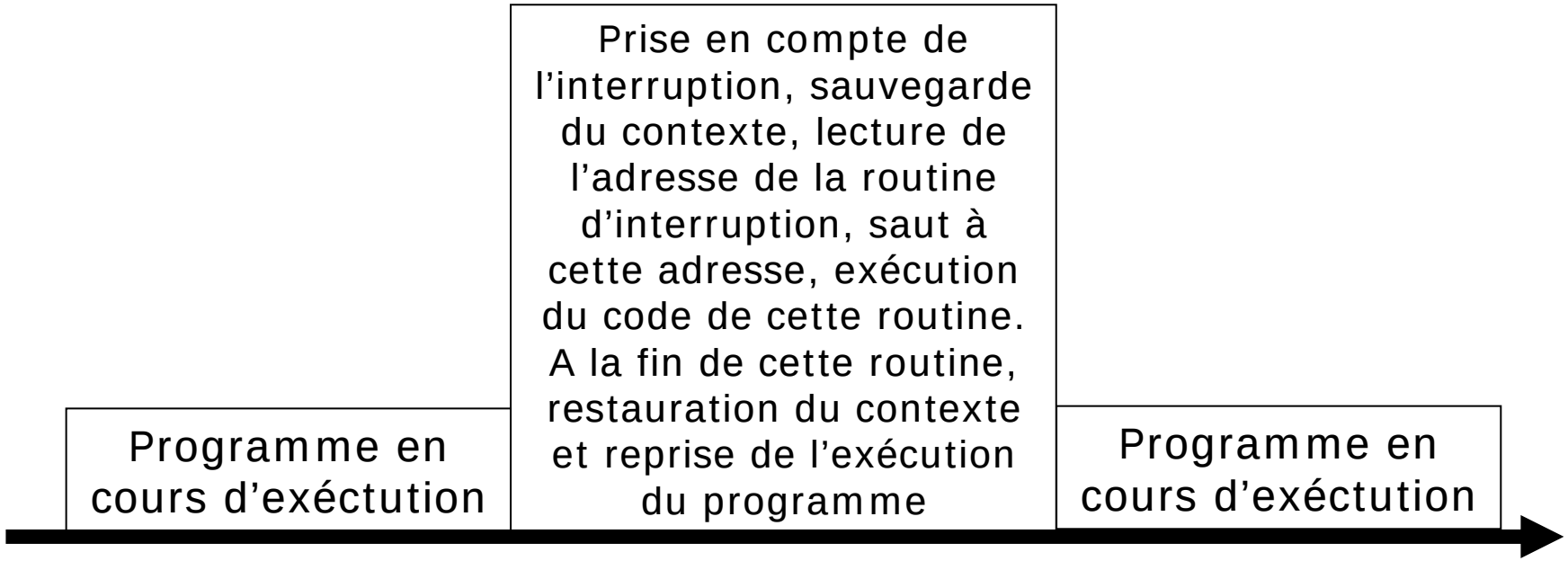
Mécanisme d'interruption

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.

Les systèmes d'exploitation

Mécanisme d'interruption



Les systèmes d'exploitation

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.

Les systèmes d'exploitation

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 :

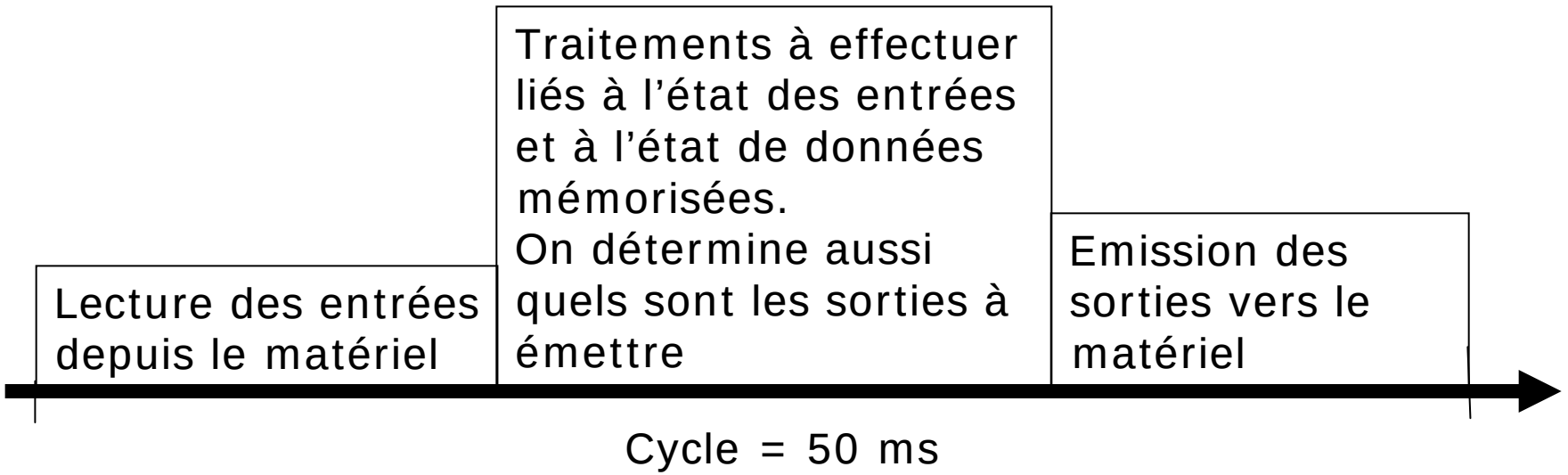
1)As-tu fini d'imprimer ?
2)Petit travail utile
3)Retour en 1.

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).

Les systèmes d'exploitation

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), Crotale NG (Thomson).

Les systèmes d'exploitation

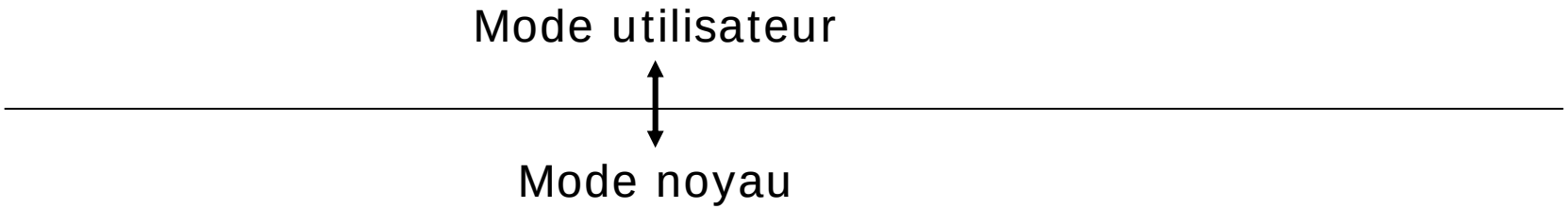
Notion système

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 matériel, 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 systèmes d'exploitation

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.
- Etc..

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 systèmes d'exploitation

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.

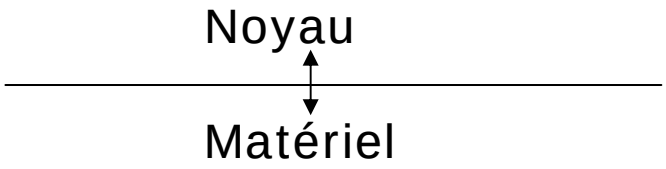
A cela peut s'ajouter des Routines de contrôle du pilote offrant, par l'intermédiaire de messages connus de l'application, des services autres que ceux ci-dessus tel que :

- Annulation de la dernière commande
- Ré – initialisation du périphérique
- Etc…

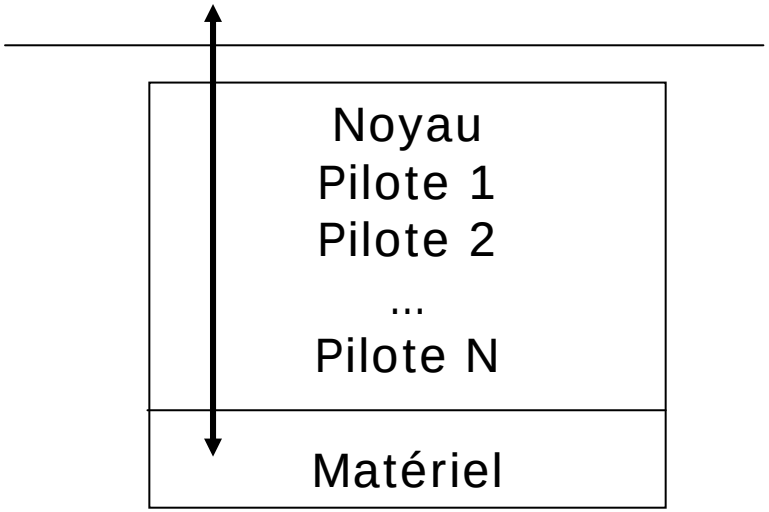
Les systèmes d'exploitation

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.

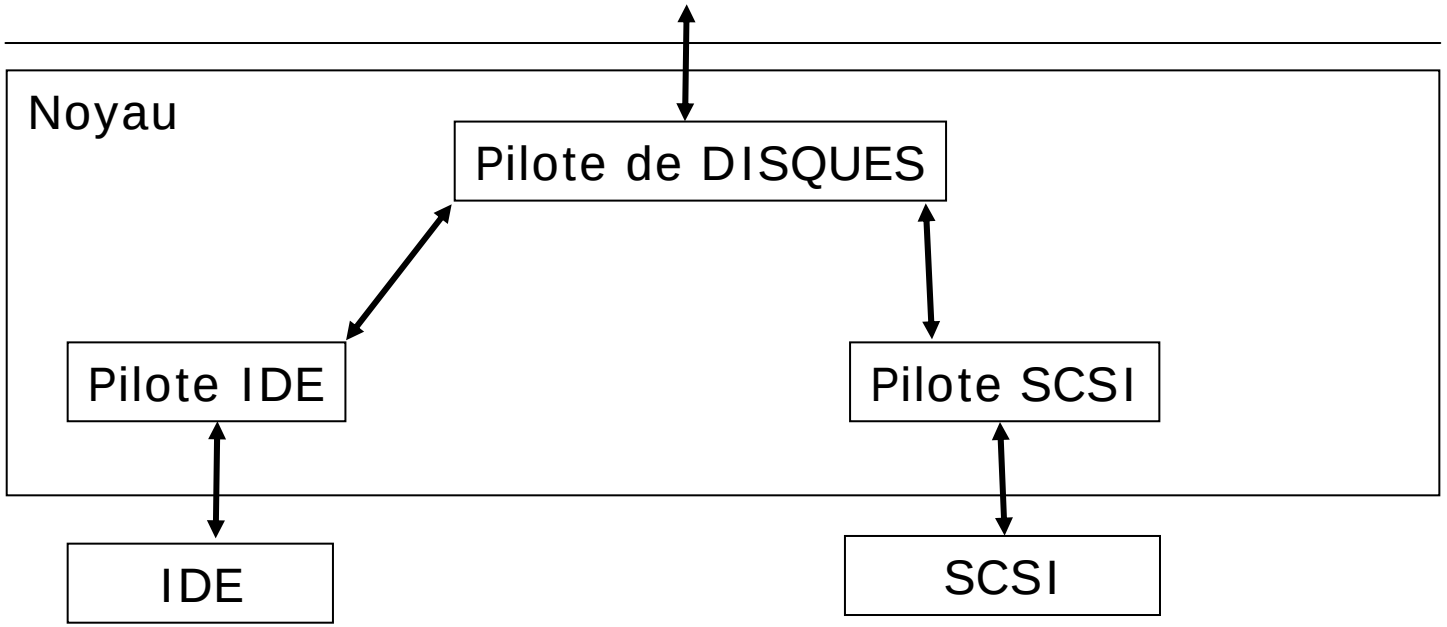
Les systèmes d'exploitation

Exemple de pilote multicouche : les disques

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.

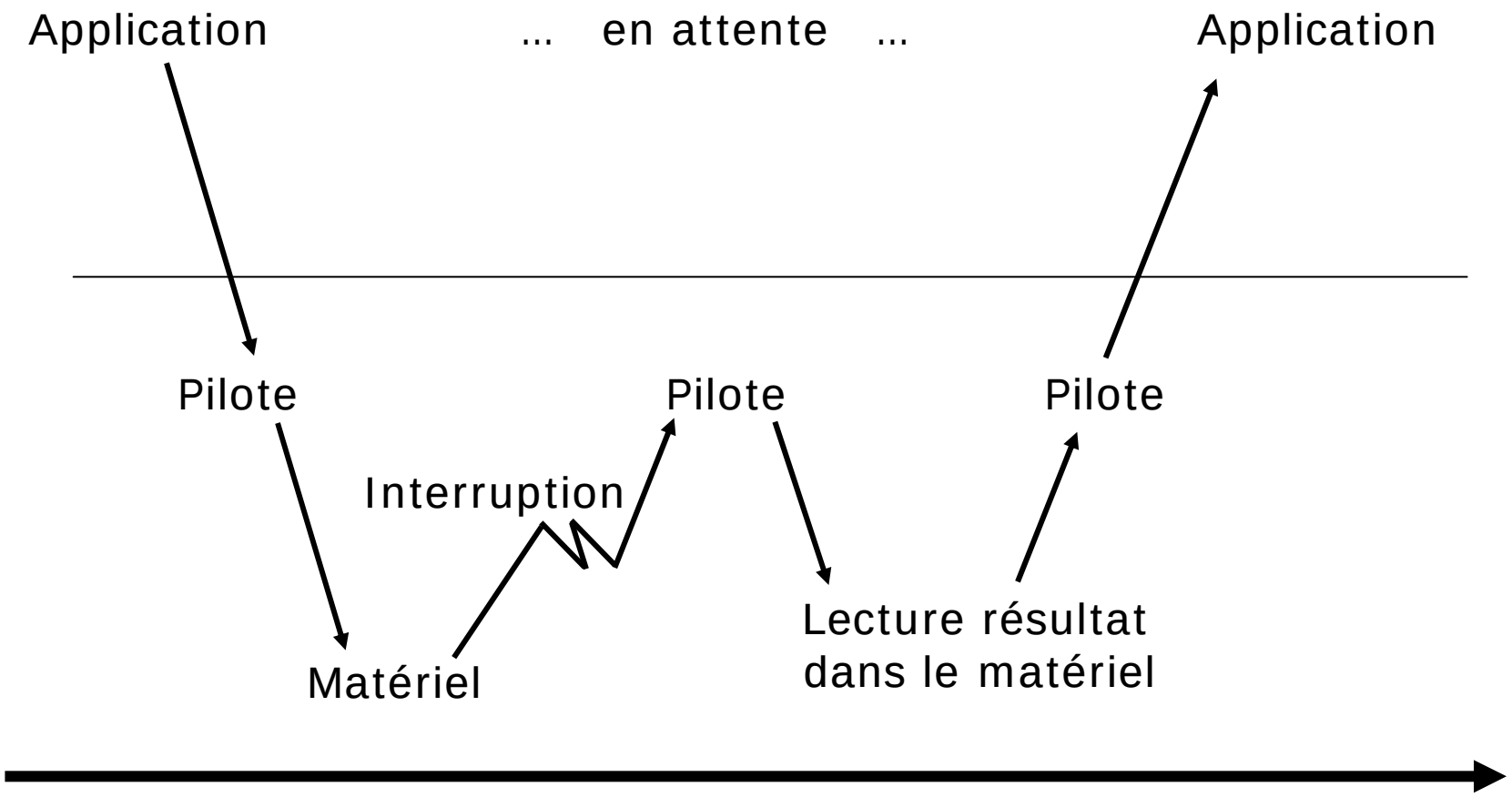
Ce pilote identifie le type de disque et effectue le travail demandé par l'intermédiaire d'un autre pilote lié au type de disques.



Les systèmes d'exploitation

Pilotes synchrones

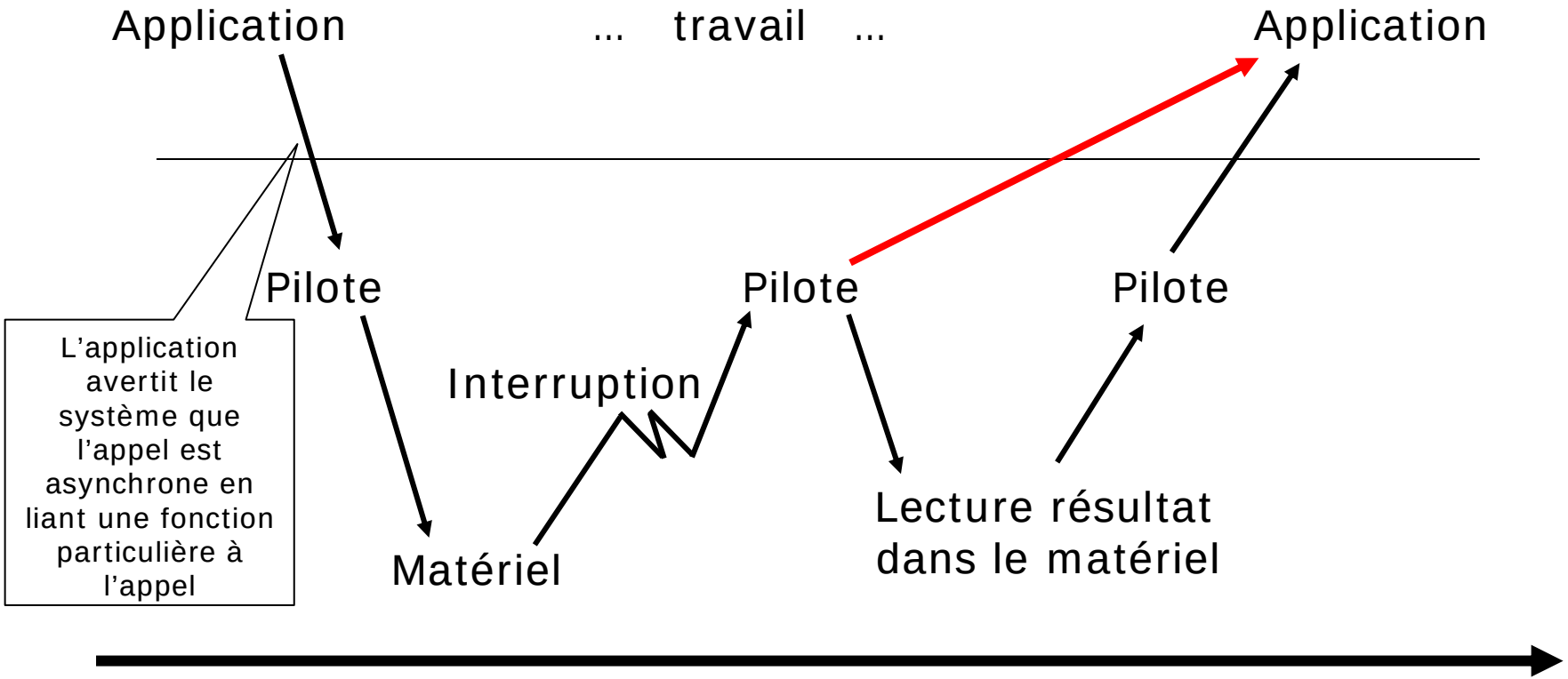
Lorsqu'une application doit attendre le résultat d'un travail demandé à un périphérique, on dit que le pilote est synchrone :



Les systèmes d'exploitation

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 averti du résultat du service demandé par une routine spéciale de l'application liée lors de l'appel système.

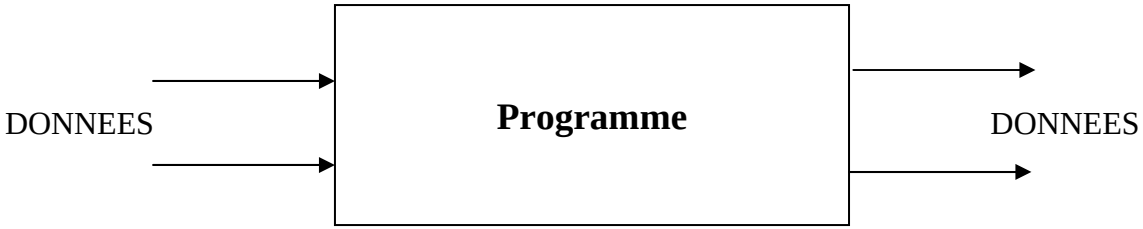


Pour réaliser cela, l'application se sert d'une routine (APC asynchronous procedure call sous windows, un masque de signaux sous unix) qui l'avertira et lira le résultat du travail demandé.

Les systèmes d'exploitation

Définition d'un programme informatique

Un programme informatique :



Un programme informatique peut être vus comme une fonction manipulant des données présentes sur son entrée et émettant des données sur sa sortie.

Les systèmes d'exploitation

Le programme est composé de code binaire

➢ Un programme est composé d'une suite d'états électriques provoquant des traitements logiques au niveau de l'unité d'exécution.

Par définition, l'être humain adopte une notation composée de 0 et de 1 suivant la position de ces états.

➢ Une instruction est donc une suite de 0 et de 1 comprenant à la fois l'instruction et ses données :

01101101 10110010

Le codage de l'instruction est dépendante du processeur et c'est le constructeur qui le définit.

L'ensemble des instructions ainsi défini forme le jeu d'instructions du processeur. Un processeur est compatible avec un deuxième processeur lorsque les deux jeux d'instructions ont le même codage, par exemple les Intel et Amd.

➢ Un programme informatique est donc une suite de 0 et de 1 représentant les différentes instructions a exécutées dans l'ordre d'apparition. C'est le langage machine.

0011010100100100101111011010101111001010001001000111111……

Les systèmes d'exploitation

Le langage assembleur

Une suite d'instructions en langage machine étant incompréhensible pour un être humain, l'homme se sert d'un programme informatique qui traduit des mnémoniques en langage machine :

Exemple extrait du jeu d'instructions du 8086 :

OR destination, source

Destination = Destination OU source, CF = 0, OF = 0

Indicateurs affectés : CF, OF, PF, SF, ZF, Indicateur non affecté : AF

Description :

OR effectue le « OU inclusif » logique des deux opérandes (octet ou mot) et renvoie le résultat dans l'opérande destination. Un bit du résultat est mis à 1 si l'un ou l'autre (ou les deux) des bits correspondant des opérandes d'origine est à 1, sinon le bit est mis à 0.

Codage : 00001101 donnée donnée

Ce jeu d'instruction est donné par le constructeur du microprocesseur pour chaque instruction.

Les systèmes d'exploitation

Le langage assembleur

Le programme traduisant les mnémoniques en langage machine s'appelle **un compilateur.**

Le compilateur doit :

- vérifier la syntaxe du langage assembleur
- vérifier le lexique du langage
- générer une suite d'instructions pour le microprocesseur.

Il connaît donc l'ensemble des instructions, il est capable :

- d'ordonnancer les instructions comme indiqué dans le code source
- de gérer les adresses mémoire
- d'établir le format du programme.

La suite d'instructions générée est mise dans un fichier s'appelant le **fichier objet**. Chaque fichier source, faisant partie d'un programme génère un fichier objet.

Les systèmes d'exploitation

Les langages haut niveau

Le langage assembleur n'est pas très performant car :

- Chaque instruction ne fait qu'une petite chose.
- Il n'y a pas de fonctions, ni de procédures.
- Les instructions interfacent directement le matériel, il n'existe donc pas de procédure d'écriture, de lecture de fichier.

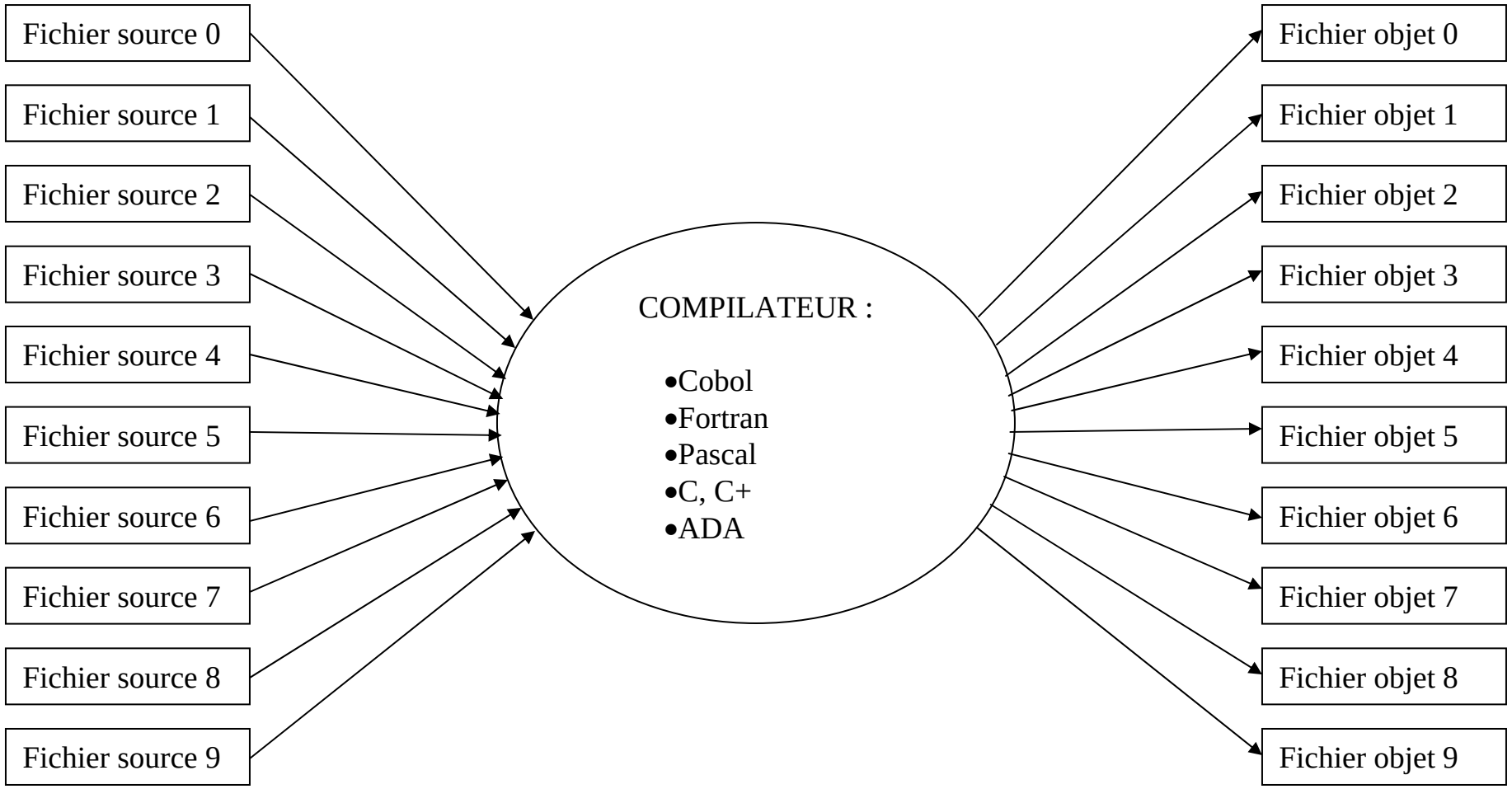
Les programmeurs ont donc créé au fil des ans d'autres langages reposant toujours sur le même principe :

- Cobol.
- Fortran.
- C.
- Ada.
- Pascal.
- C++

Les systèmes d'exploitation

Les langages haut niveau

Tout comme l'assembleur, ces langages génèrent un fichier objet par fichier source :



Les systèmes d'exploitation

Les langages haut niveau

Un fichier objet peut contenir une référence à un autre fichier objet :

Fichier fact.c :

```
int factorielle (int i)
{
  int a ;

  a = i ;
  if ( a== 1)
     return 1 ;
  else
    return i * factorielle( a - 1) ;
}
```

Fichier main.c :

```
#include <stdio.h>
extern int factorielle( int i);

int v ;

void main( void)
{

   v = factorielle(5) ;
   printf("La factroielle de 5 est egale a %d\n", v) ;
}
```

fichier fact.o (pseudo format) :

Nom du module : factorielle, adresse @f
Paramètre entrée : int i, adresse @i
Paramère retour : int adresse @retour
Variable locale : int a, adresse @a
Point d'entrée : @entrée

Début (@f)
[Réservation dans la pile des variables locales]
[Lecture depuis la pile des paramètres]

points d'entrées:
…
… code de la fonction
…

[Ecriture dans la pile des paramètres de sortie]
retour à l'appelant (@sortie)

fichier main.o (pseudo format) :

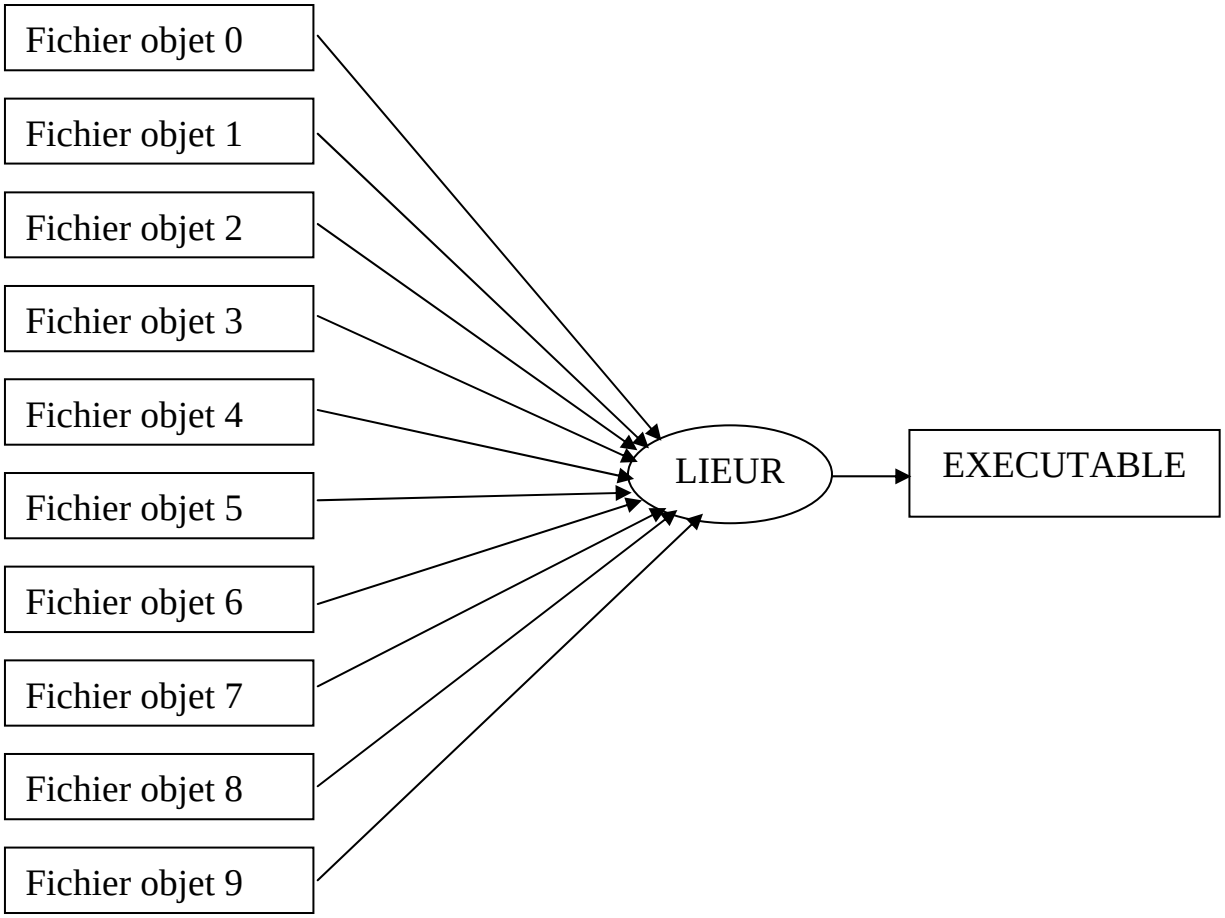
Nom du module : main
Paramètre entrée : néant
Paramètre retour : néant
Variable locale : néant
Point d'entrée : @entrée

Début (@0)
Variable globale entière @v, 32 bits.
Fonction externe @f, printf @p
Point d'entrée (@entrée = @0 + 4)
…
… code de la fonction
Appelle factorielle, @f
…
appelle printf, @p
Point de sortie du programme

Les systèmes d'exploitation

Les langages haut niveau

L'ensemble des fichiers objets sont alors passés dans un lieur (linker) pour obtenir un exécutable.



Le lieur ne résolve que les appels inter-modules.

Nom du module : factorielle, adresse @f
Paramètre entrée : int i, adresse @i
Paramère retour : int adresse @retour
Variable locale : int a, adresse @a
Point d'entrée : @entrée

Début (@0)
[Réservation dans la pile des variables locales]
[Lecture depuis la pile des paramètres]

points d'entrées:
…
… code de la fonction
…

[Ecriture dans la pile des paramètres de sortie]
retour à l'appelant (@sortie)

Nom du module : main
Paramètre entrée : néant
Paramètre retour : néant
Variable locale : néant
Point d'entrée : @entrée

Début (@0)
Variable globale entière @v, 32 bits.
Fonction externe @f, printf @p
Point d'entrée (@entrée = @0 + 4)
…
… code de la fonction
Appelle factorielle, @f
…
appelle printf, @p
Point de sortie du programme

Appelle librairie dynamique

Nom du module : main
Paramètre entrée : néant
Paramètre retour : néant
Variable locale : néant
Point d'entrée : @entrée

Début (@0)
Variable globale entière @v, 32 bits.

Début factorielle (@f = @0+Y) :
[Réservation dans la pile des variables locales]
[Lecture depuis la pile des paramètres]

début code…
… code de la fonction
…

[Ecriture dans la pile des paramètres de sortie]
retour à l'appelant (@sortie = @0+n)

externe printf @p
Point d'entrée (@entrée = @0 + 4)
…
… code de la fonction
Appelle factorielle, @f
…
appelle printf, @p
Point de sortie du programme

Appelle librairie statique

Nom du module : main
Paramètre entrée : néant
Paramètre retour : néant
Variable locale : néant
Point d'entrée : @entrée

Début (@0)
Variable globale entière @v, 32 bits.

Début printf
[Réservation dans la pile des variables locales]
[Lecture depuis la pile des paramètres]

points printf(@p = @0 + m) :
…
… code de la fonction
…

[Ecriture dans la pile des paramètres de sortie]
retour à l'appelant (@sortie)

Début factorielle (@f)
[Réservation dans la pile des variables locales]
[Lecture depuis la pile des paramètres]

points d'entrées (@entrée) :
…
… code de la fonction
…

[Ecriture dans la pile des paramètres de sortie]
retour à l'appelant (@sortie)
Point d'entrée (@entrée = @0 + 4)
…
… code de la fonction
Appelle factorielle, @f
…
appelle printf, @p
Point de sortie du programme

Les systèmes d'exploitation

Chargeur du système d'exploitation

Rôle du chargeur dans le système d'exploitation :

- Le chargeur d'un système d'exploitation doit réserver la mémoire pour le chargement du fichier exécutable.
- Résoudre les adresses d'appels interne dans le fichier exécutable au moment où le fichier est transférer en mémoire.
- Résoudre les adresses d'appels aux librairies dynamique au moment du chargement.
- Effectuer un saut au point d'entrée du programme.

Appelle librairie dynamique (chargement en 315)

315 : Variable globale entière @v, 32 bits.
316
317
318
319 : Début factorielle (@f = @0+Y) :
320 Réservation et élaboration des
321 variables locales dans la pile
322
323
324 Repérage des paramètres dans la pile
325
326 Début code de la fonction
327
328
329
330
331
332 Ecriture des résultats dans la pile
333
334 Dépilage de l'adresse de retour et retour

335 Point d'entrée du programme
336 Début code main
337 Enpilage de v
338 Appel @f
339 Dépilage du résultat et
340 affectation du résultat à v
341 Enpilage de v
342 appel fonction printf en librairie dynamqie @4045
343 sortie du programme