Processus concurrents et parallélisme

Chapitre 1 - Notions de base

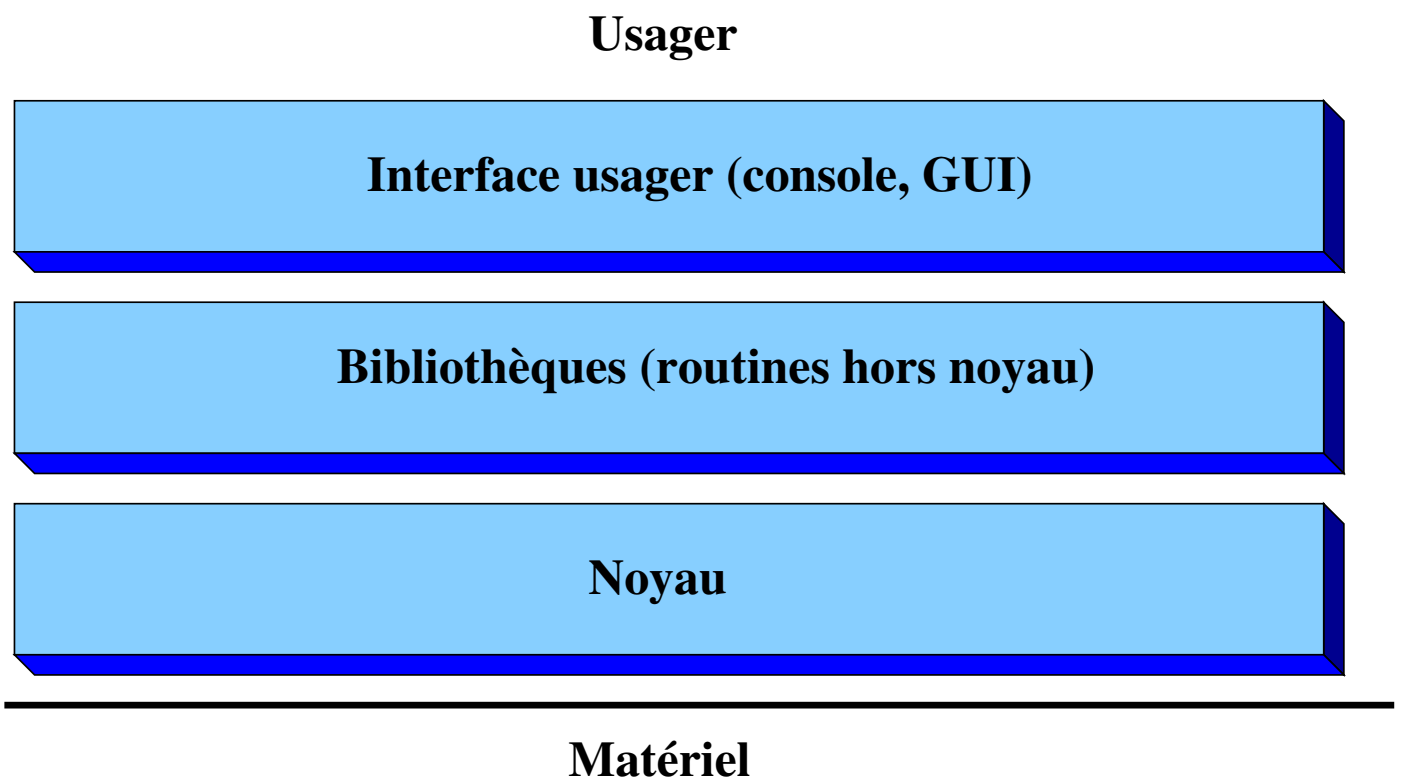
Gabriel Girard

5 janvier 2023

Chapitre 1 - Notions de base

Chapitre 1 - Notions de base

Structure OS



Pourquoi ?

- Implante certains outils pour le parallélisme.
  Processus, fils d'exécution, sémaphores, mutex, verrous, ...
- Est un logiciel parallèle.
  Problèmes de synchronisation et d'interblocage.

Qu'est-ce qu'un noyau de système d'exploitation ?

- Coeur du système d'exploitation
- Représente une petite partie du système mais la plus utilisée
- Réside en grande partie en permanence en mémoire centrale

Les principales fonctions d'un noyau sont :

- Gestion des interruptions
- **Gestion des processus et fils d'exécution**
- Gestion de l'UCT et du temps
- Gestion de la mémoire
- **Gestion de la synchronisation**
- **Gestion de la communication**
- Gestion des blocs de contrôle
- Gestion des fichiers (*)
- Gestion des activités d'E/S (*)
- Gestion des codes d'accès (*)

Les principales primitives (API) d'un noyau sont :

- Certaines fonctions offrent des primitives :
  - Obtenir la date
  - Allouer/libérer de la mémoire
  - Créer/Détruire des processus et fils d'exécution (+)
  - Synchronisation (Wait, Post, P, V, Lock, Unlock, ...)
  - Communication (send, receive, ...)
- D'autres pas (interruptions et blocs de contrôle)

Qu'est-ce qu'un micro-noyau ?

- On retire le plus de fonctions possibles du noyau
  - gestion des fichiers
  - gestion des codes d'accès
  - gestion de la pagination et du swapping
  - ...
- Exemples : Mach (OS/X), Qnx, Hurd, Amoeba, Zircon, ...

Pourquoi un micro-noyau ?

- Modularité
- Flexibilité
- Fiabilité
  - Selon certaines études
    $\rightarrow$ entre 2 et 10 erreurs / 1000 lignes de code

Pourquoi un micro-noyau ?

- Modularité
- Flexibilité
- Fiabilité
  - Selon certaines études
    $\rightarrow$ entre 2 et 10 erreurs / 1000 lignes de code
  - Un OS monolithique contenant 5 millions de lignes de code
    $\rightarrow$ entre 10 000 et 50 000 erreurs

Pourquoi un micro-noyau ?

- Modularité
- Flexibilité
- Fiabilité
  - Selon certaines études
    $\rightarrow$ entre 2 et 10 erreurs / 1000 lignes de code
  - Un OS monolithique contenant 5 millions de lignes de code
    $\rightarrow$ entre 10 000 et 50 000 erreurs
  - Linux (2018) $\equiv$ 25 millions de lignes de code
    $\rightarrow$ entre 50 000 et 250 000 erreurs

Pourquoi un micro-noyau ?

- Modularité
- Flexibilité
- Fiabilité
  - Selon certaines études
    $\rightarrow$ entre 2 et 10 erreurs / 1000 lignes de code
  - Un OS monolithique contenant 5 millions de lignes de code
    $\rightarrow$ entre 10 000 et 50 000 erreurs
  - Linux (2018) $\equiv$ 25 millions de lignes de code
    $\rightarrow$ entre 50 000 et 250 000 erreurs
  - Windows $\equiv$ 50 millions de lignes de code
    $\rightarrow$ entre 100 000 et 500 000 erreurs

Caractéristiques d'une implantation d'un noyau

- Se doit d'être fiable
  - pas de problème de synchronisation ni d'interblocage
  - pas d'accès direct
- Doit donc être isolé de toutes influences externes directes

Fiabilité : isolation

- Accès via API (ensemble de primitives)
- Exemples d'API : Win32 (Win64), Unix, Posix

- API implantée via interruptions logiques (appels systèmes)
- Le système fournit un pilote qui valide chaque appel et l'oriente vers la bonne fonction

Fiabilité : concurrence

- Comment faire la synchronisation ?

Fiabilité : concurrence

- Comment faire la synchronisation ?
  outils matériels, désactivation des interruptions

Fiabilité : concurrence

- Comment faire la synchronisation ?
  outils matériels, désactivation des interruptions
- Comment éviter les interblocages ?

Fiabilité : concurrence

- Comment faire la synchronisation ?
  outils matériels, désactivation des interruptions
- Comment éviter les interblocages ?
  méthodes de conception, ...

Désactivation des interruptions ?

Pour des raisons de fiabilité (et de synchronisation), le noyau doit périodiquement désactiver les interruptions

Désactivation des interruptions ?

Pour des raisons de fiabilité (et de synchronisation), le noyau doit périodiquement désactiver les interruptions

Problème possible :

désactivation trop longue

Conception du noyau

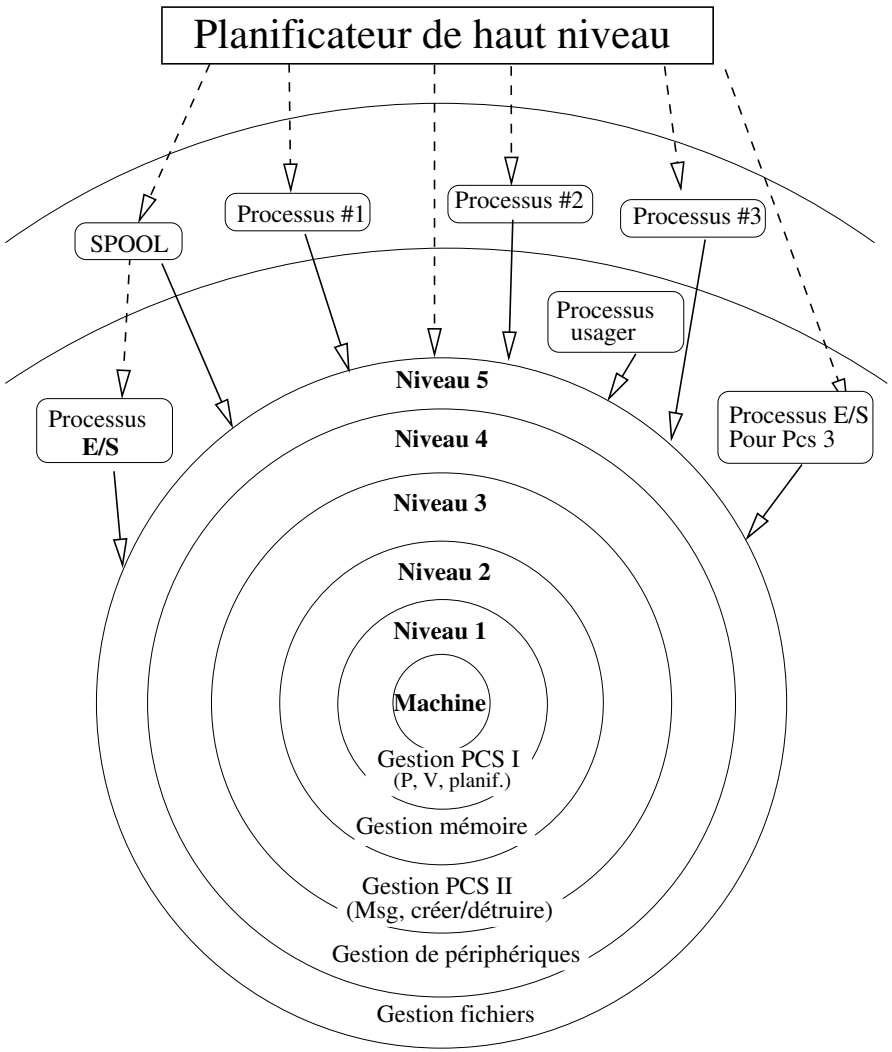
- Pour des raisons de fiabilité, le noyau est conçu de façon hiérarchique (évite les interblocages)
  - Niveau 0 $\rightarrow$ la machine
  - Niveau 1 $\rightarrow$ fonction 1 (gestion UCT)
    Au dessus de cette fonction on a une machine virtuelle (plusieurs UCTs virtuelles)
  - ...
  - Niveau N $\rightarrow$ dernière couche (noyau complet)

- Niveau *i* ne peut faire appel qu'aux niveaux inférieurs

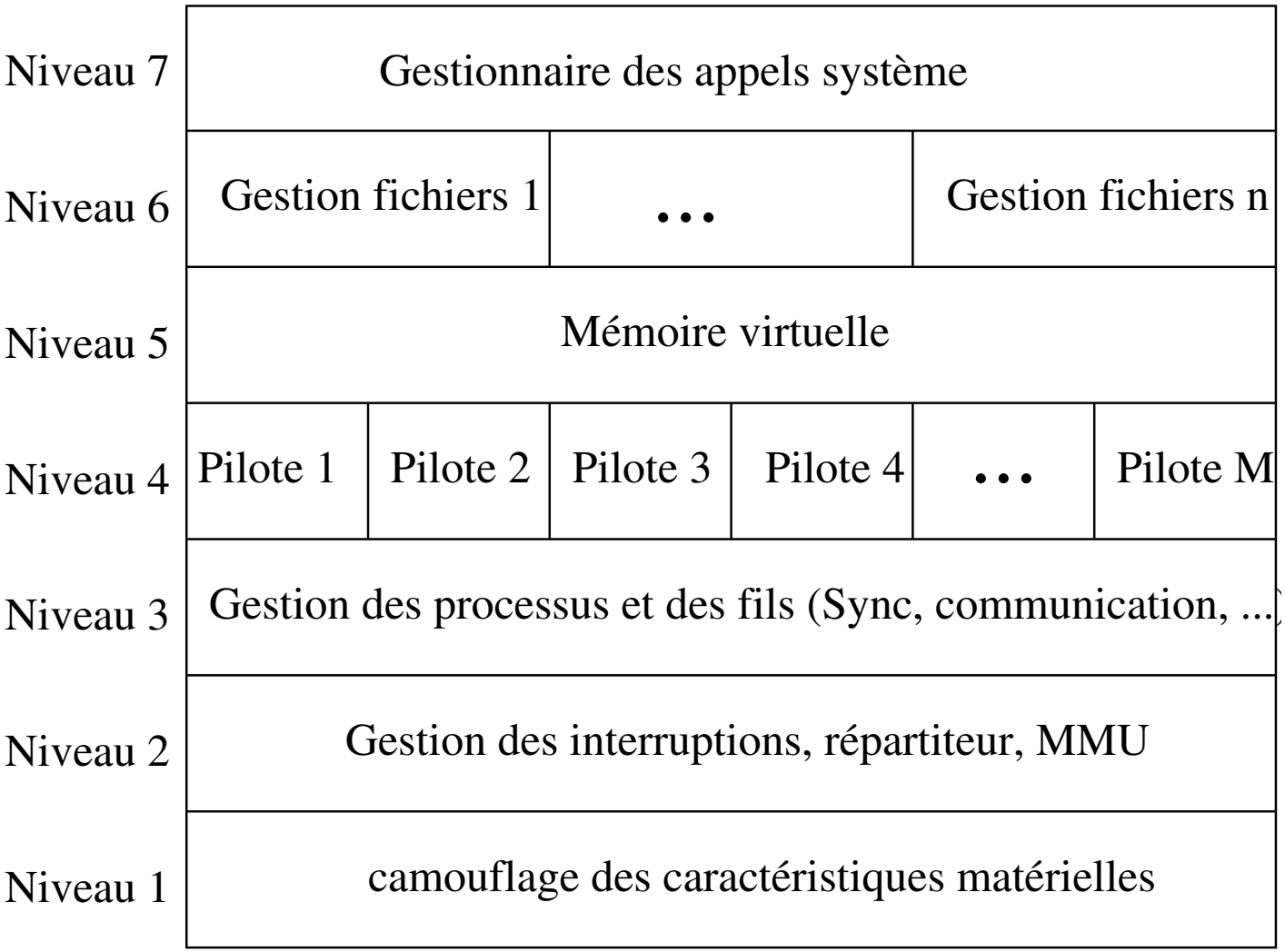
Avantages et inconvénients

- Avantages et inconvénients ? ? ?
- La conception hiérarchique s'applique aussi à l'extérieur du noyau (hiérarchie de processus système)

Exemple


Planificateur de haut niveau
SPOOL
Processus #1
Processus #2
Processus #3
Processus usager
Processus E/S
Processus E/S Pour Pcs 3
Niveau 5
Niveau 4
Niveau 3
Niveau 2
Niveau 1
Machine
Gestion PCS I
(P, V, planif.)
Gestion mémoire
Gestion PCS II
(Msg, créer/détruire)
Gestion de périphériques
Gestion fichiers

Exemple


Niveau 7
Gestionnaire des appels système
Niveau 6
Gestion fichiers 1
...
Gestion fichiers n
Niveau 5
Mémoire virtuelle
Niveau 4
Pilote 1
Pilote 2
Pilote 3
Pilote 4
...
Pilote M
Niveau 3
Gestion des processus et des fils (Sync, communication, ...)
Niveau 2
Gestion des interruptions, répartiteur, MMU
Niveau 1
camouflage des caractéristiques matérielles

Chapitre 1 - Notions de base

Qu'est-ce qu'un processus ? ?

Qu'est-ce qu'un processus ? ?

- Programme en exécution

Qu'est-ce qu'un processus ? ?

- Programme en exécution
- Possède un état

Qu'est-ce qu'un processus ? ?

- Programme en exécution
- Possède un état
- Un programme n'est pas un processus

Qu'est-ce qu'un processus ? ?

- Programme en exécution
- Possède un état
- Un programme n'est pas un processus
- Entité dynamique qui naît et qui meurt

Qu'est-ce qu'un processus ? ?

Un processus est un type abstrait

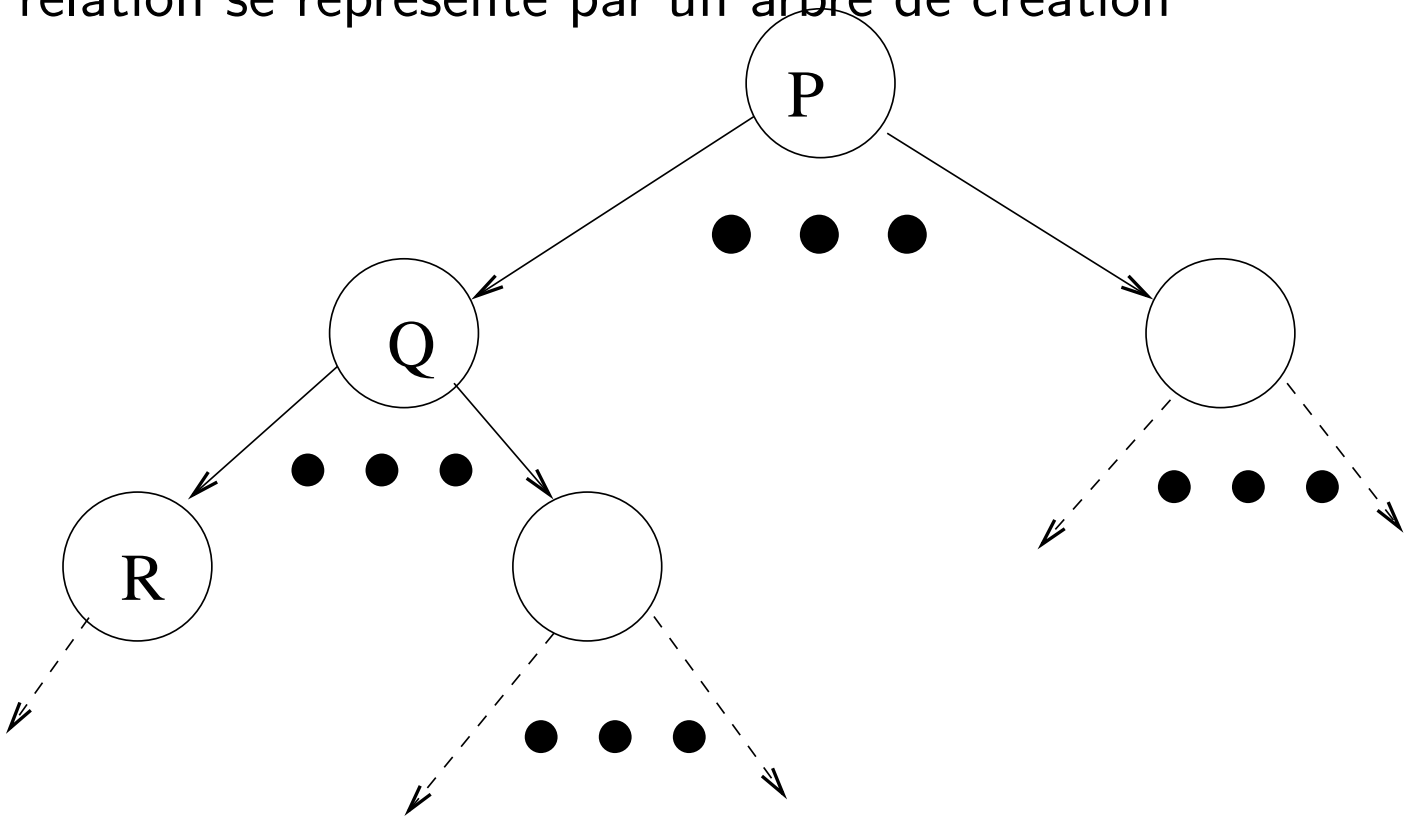
- Données
- Opérations

Liens entre les processus

- Un processus est généralement créé par un autre
- Relation parent/enfant
  - Un processus Q créé par P appartient à la descendance de P
  - Tout processus créé par Q appartient à la descendance de P

Liens entre les processus

Cette relation se représente par un arbre de création



Création

La création implique :

- l'assignation d'un nom
- la création d'un bloc de contrôle
- le démarrage synchrone ou asynchrone
- le partage total, partiel ou nul des ressources
- l'allocation d'un pouvoir initial
- l'allocation des liens de communication

Destruction

La destruction :

- peut se faire de trois façons
- affecte la descendance
- implique un nettoyage

Autres opérations

- suspendre/poursuivre
- démarrer/arrêter
- ...

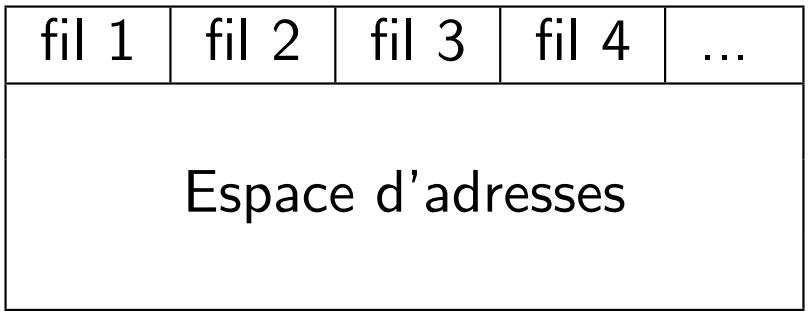
Exemples d'opérations

- Unix : fork/join, execl, ...
- Windows : NTcreateProcess, NTcreateThread

Chapitre 1 - Notions de base

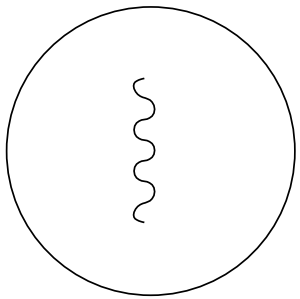
Qu'est-ce qu'un fil d'exécution ?

- Plusieurs exécutions dans un même espace d'adresses (processus légers)



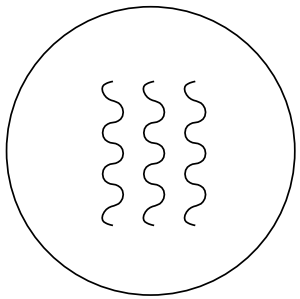
Fil d'exécution vs processus

(a)



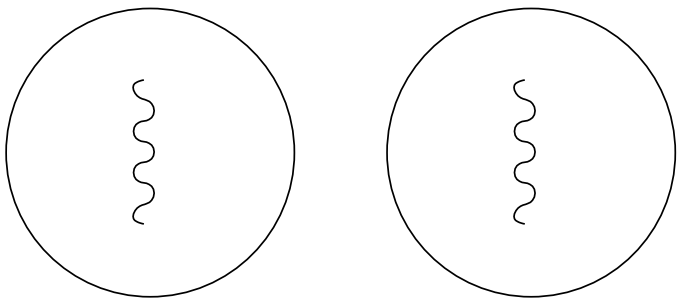
Un processus
Un fil d'execution

(c)



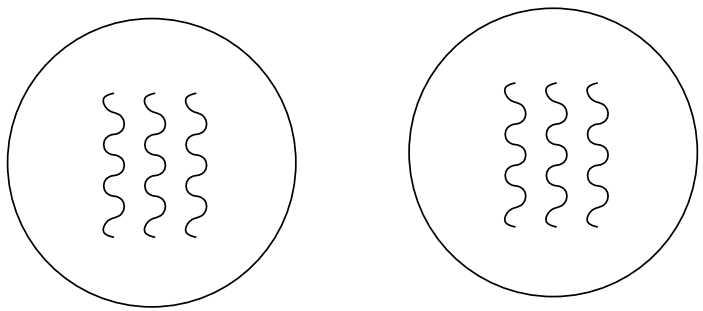
Un processus
Plusieurs fils d'execution

(b)



Plusieurs processus
Un fil d'execution chacun

(d)



Plusieurs processus
Plusieurs fils d'execution

Structure des fils d'exécution

Un ensemble de fils d'exécution partagent :

- espace d'adresses (code et données)
- ressources

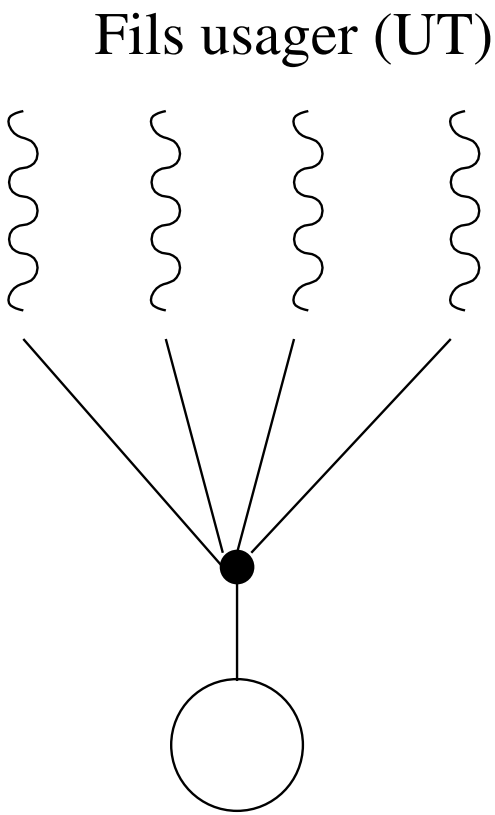
Un ensemble de fils d'exécution ne partagent pas :

- les registres
- la pile

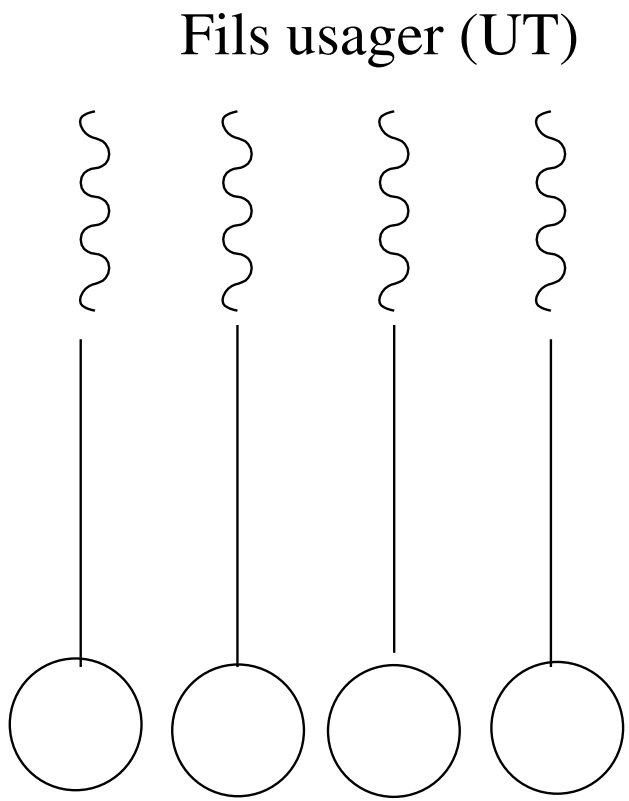
Terminologie, utilité et types

- Tâche $\rightarrow$ environnement d'un fil
- Processus $\equiv$ tâche avec un seul fil
- Fournissent la « ré-entrance » au niveau des processus
- Deux types : noyau et usager
- Avantages des fils d'exécution par rapport au pcs ? ? ?
  - performance
  - partage

Fil de niveau usager vs fil de niveau noyau



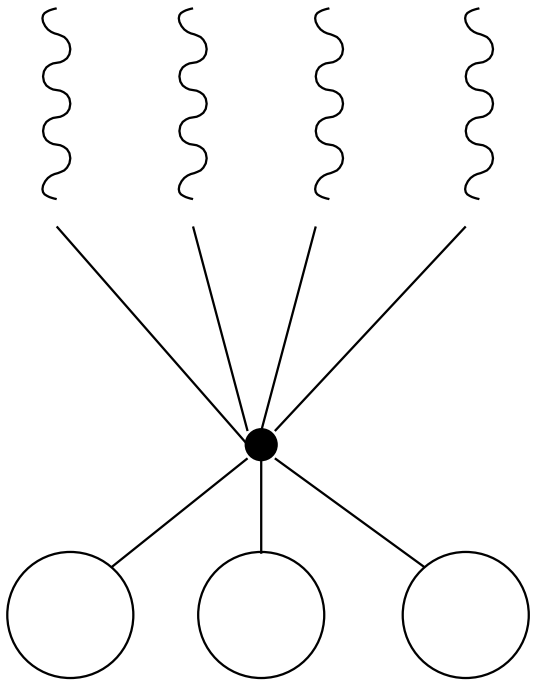
Plusieurs––un



Un––un

Fil de niveau usager vs fil de niveau noyau

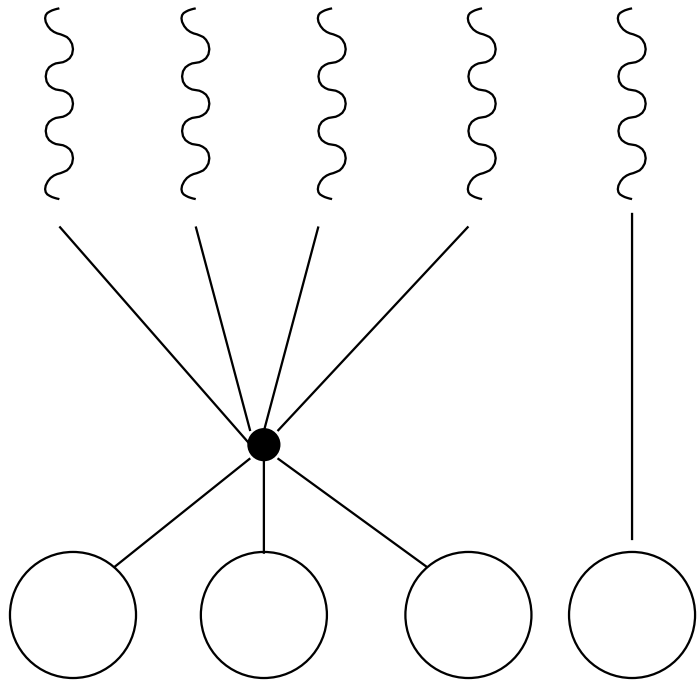
Fils usager (UT)



Fils noyau (KT)

Plusieurs--plusieurs

Fils usager (UT)



Fils noyau (KT)

Deux–niveaux

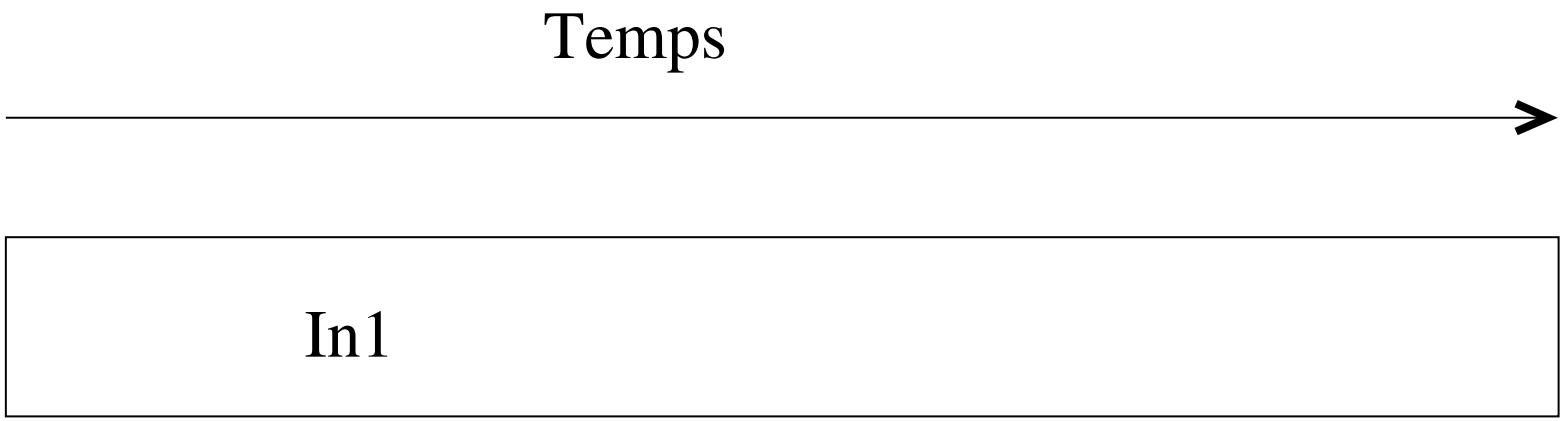
Exemples d'implantation

- Solaris : thread et lightweight process (deux-niveaux et un–un)
- Posix : thread et lightweight process (Un–un)
- Linux : thread et process (un à un)
- Windows XP et avant : thread (Un–un) et fiber (plusieurs à plusieurs)
- Windows 7 : thread (plusieurs à plusieurs)

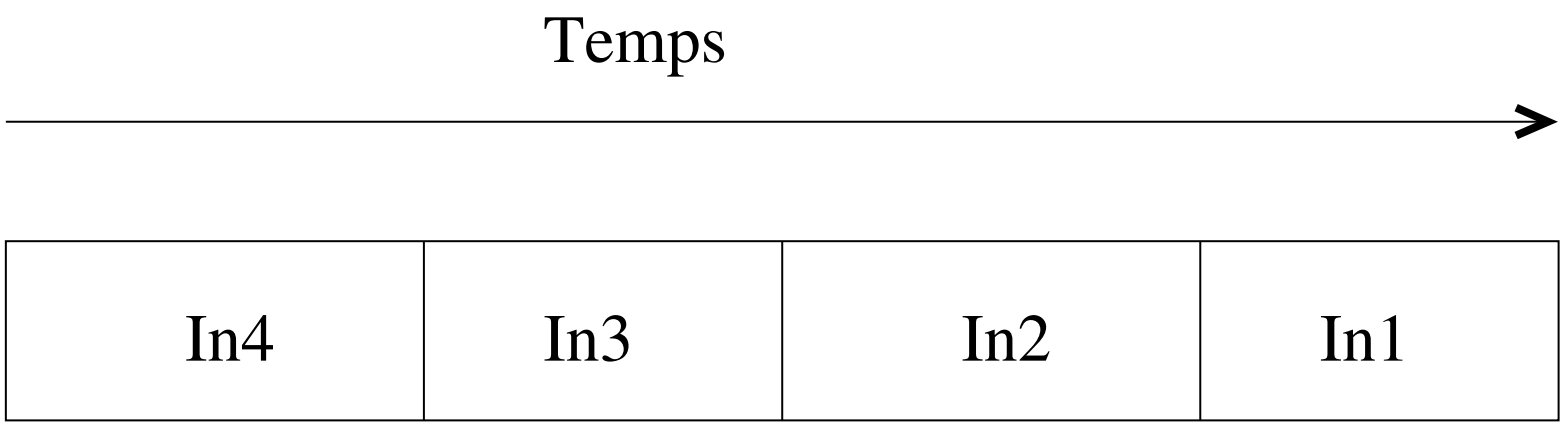
Quelques définitions

- Parallélisme au niveau des instructions (ILP)
  - Pipeline
  - Super-scalaire
  - VLIW
- Parallélisme au niveau des fils (TLP)
  - Multi-processeurs
  - Multi-coeurs
  - Multi-fils

Une seule unité d'exécution sans pipeline



Une seule unité d'exécution avec pipeline



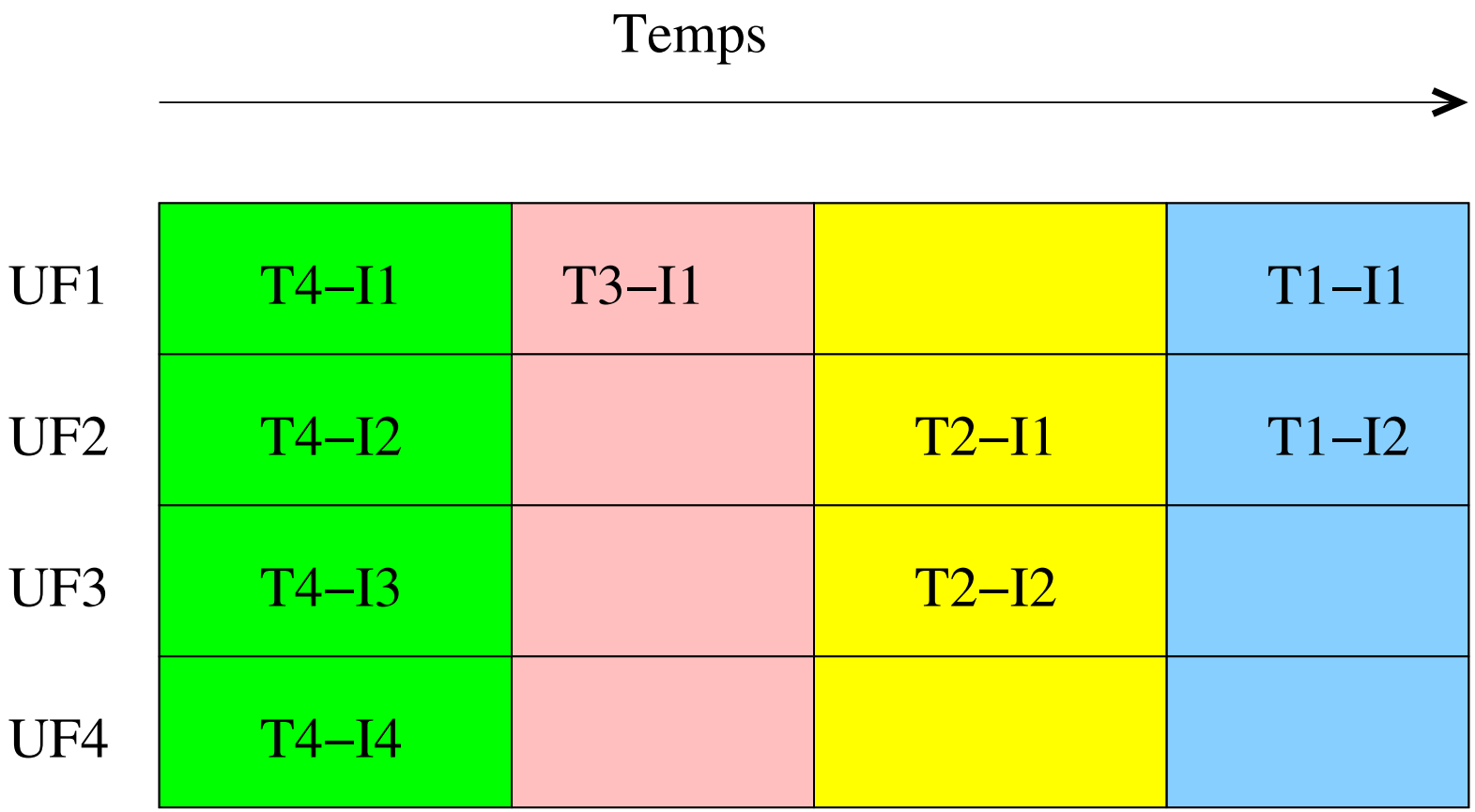
Plusieurs unités fontionnelles (super-scalaire)

UF1	In13	In9	In5	In1
UF2	In14	In10	In6	In2
UF3	In15	In11	In7	In3
UF4	In16	In12	In8	In4

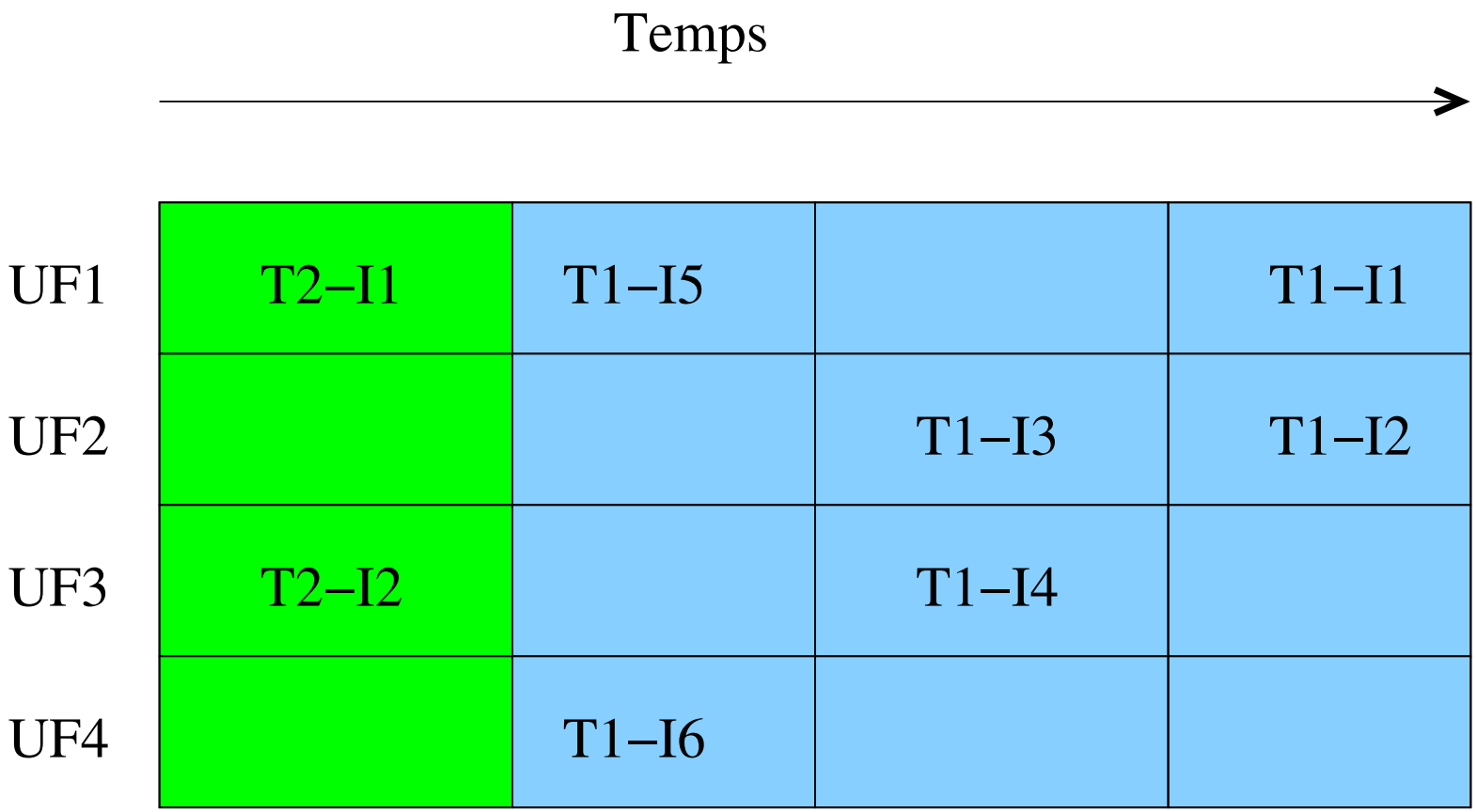
Plusieurs unités fontionnelles (super-scalaire)

UF1	In4	In3		In1
UF2	In5			In2
UF3	In6			
UF4				

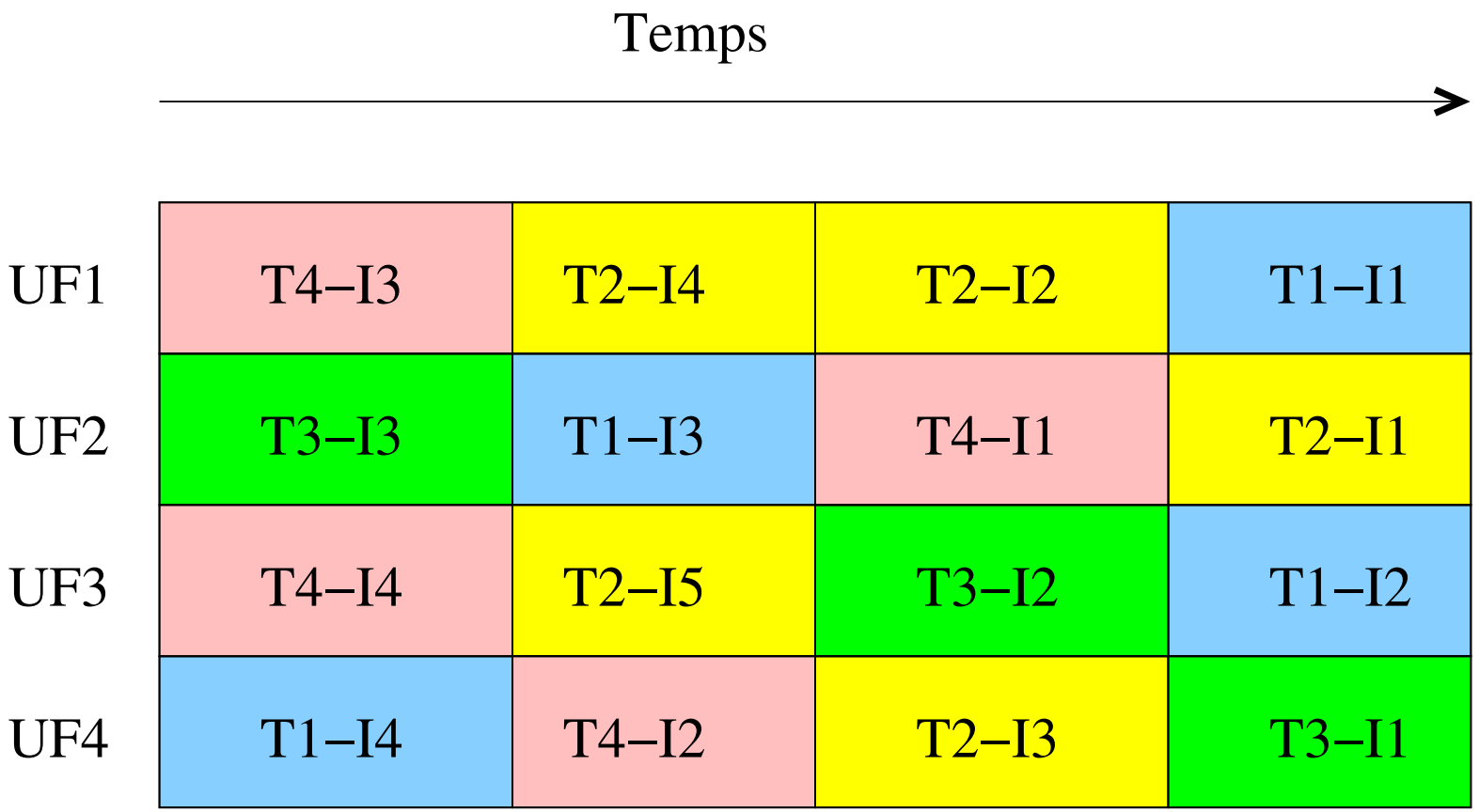
Multi-fils finement intercalés



Multi-fils grossièrement intercalés



Multi-fils simultanés



Chapitre 1 - Notions de base

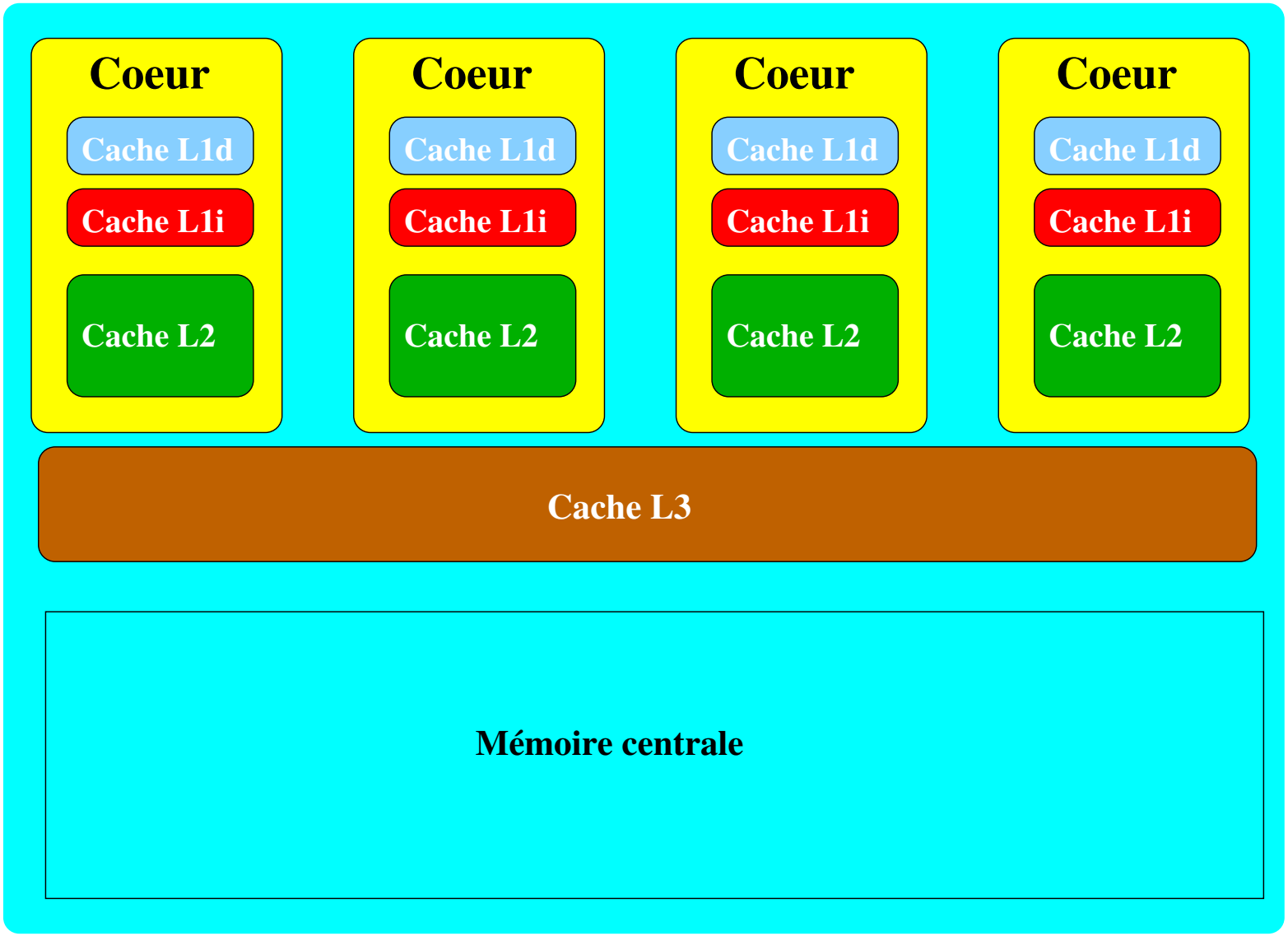
Multi-coeurs

- Généraux (2/4/8/16)
  Tous les coeurs exécutent des programmes différents
  (Intel, AMD, Sparc, Power5/6/7)
- Spécialisés (512+)
  Tous les coeurs exécutent la même fonction
  (Nvidia/Cuda/GeForce/Quadro, AMD/Radeon)

Mémoire cache

- Mémoire rapide qui emmagasine temporairement les données provenant de la mémoire centrale
- Elle affecte la performance d'un programme parallèle
- Il peut exister des caches séparées pour le code et les données.
- Il peut y avoir plusieurs niveaux de cache :
  1. L1 : intégrée au processeur (séparée D/I)
  2. L2 : normalement sur la même puce, privée ou partagée (pas de D/I)
  3. L3 : sur la même carte mère (pas de D/I)

Mémoire cache



Mémoire cache

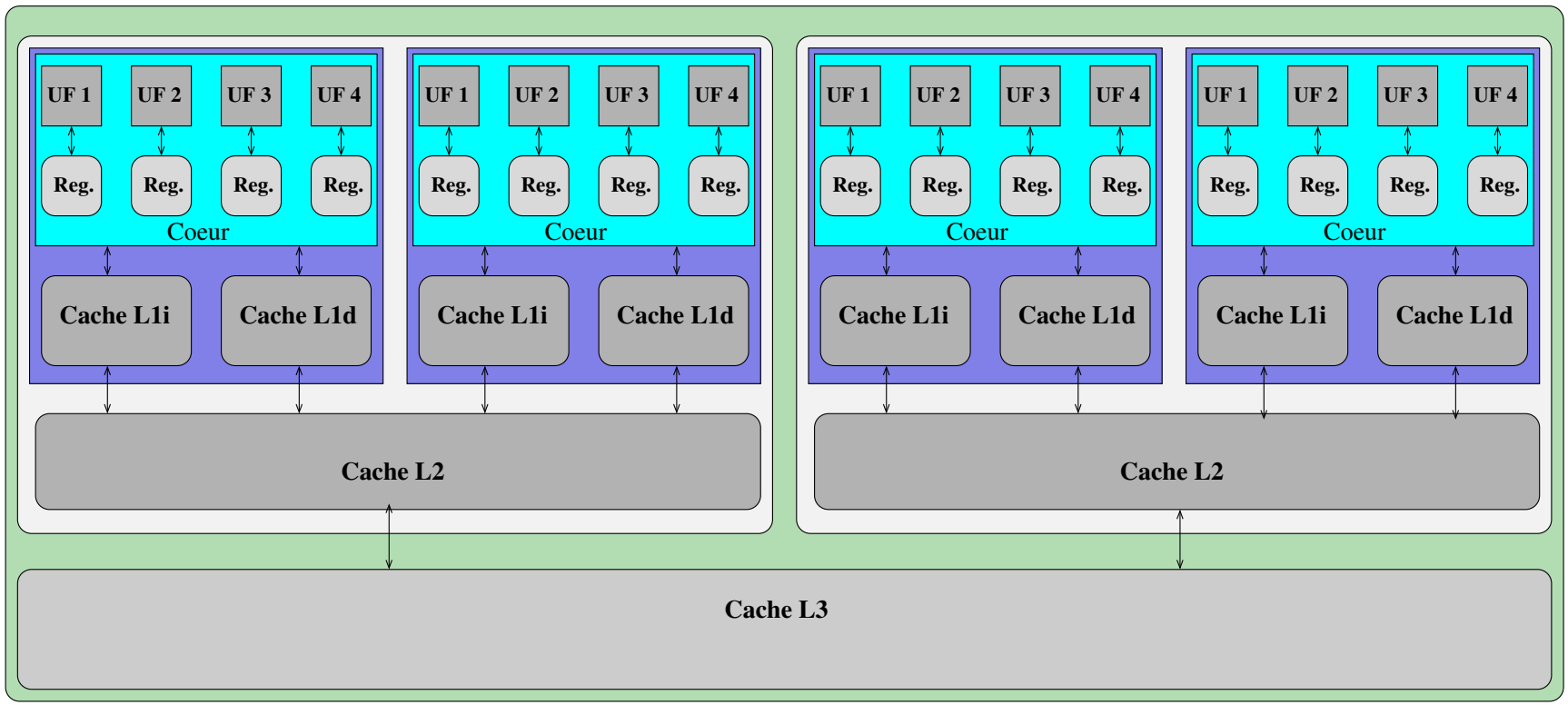


FIGURE – Structure de la mémoire cache

Mémoire cache

- On peut lire des données directement de la cache
- On peut écrire des données directement dans la cache
- Il existe plusieurs algorithmes pour synchroniser la cache et la mémoire centrale
- Il arrive que les données en cache ou en mémoire centrale soient incohérentes.