



ARCHITECTE DES SYSTÈMES D'INFORMATION

SYSTÈMES D'EXPLOITATION ET VIRTUALISATION

🎓 Mastère Spécialisé

🏛️ CentraleSupélec Exed

📅 2023/2024



Idir AIT SADOUNE

idir.aitsadoune@centralesupelec.fr

IDIR AIT SADOUNE



IDIR AIT SADOUNE

- Docteur en Informatique diplômé par l'ENSMA en 2010.



IDIR AIT SADOUNE

- Docteur en Informatique diplômé par l'ENSMA en 2010.
 - Thèse sur la modélisation et la vérification des services par une approche basée sur le raffinement et sur la preuve.





IDIR AIT SADOUNE

- **Docteur en Informatique** diplômé par l'**ENSMA** en **2010**.
 - Thèse sur la modélisation et la vérification des services par une approche basée sur le raffinement et sur la preuve.
- **Enseignant-chercheur** au sein du département informatique de **CentraleSupélec**.



IDIR AIT SADOUNE

- **Docteur en Informatique** diplômé par l'**ENSMA** en **2010**.
 - Thèse sur la modélisation et la vérification des services par une approche basée sur le raffinement et sur la preuve.
- **Enseignant-chercheur** au sein du département informatique de **CentraleSupélec**.
- **Chercheur** au sein des pôles **Modèles et Preuve** du **LMF - Laboratoire Méthodes Formelles**.

DÉMARCHE POUR ABORDER CE SUJET

DÉMARCHE POUR ABORDER CE SUJET

- Connaître les **techniques de virtualisation** :
 - proposées par un **OS**
 - proposées par un **hyperviseur**
 - proposées par les **conteneurs**
 - ...

DÉMARCHE POUR ABORDER CE SUJET

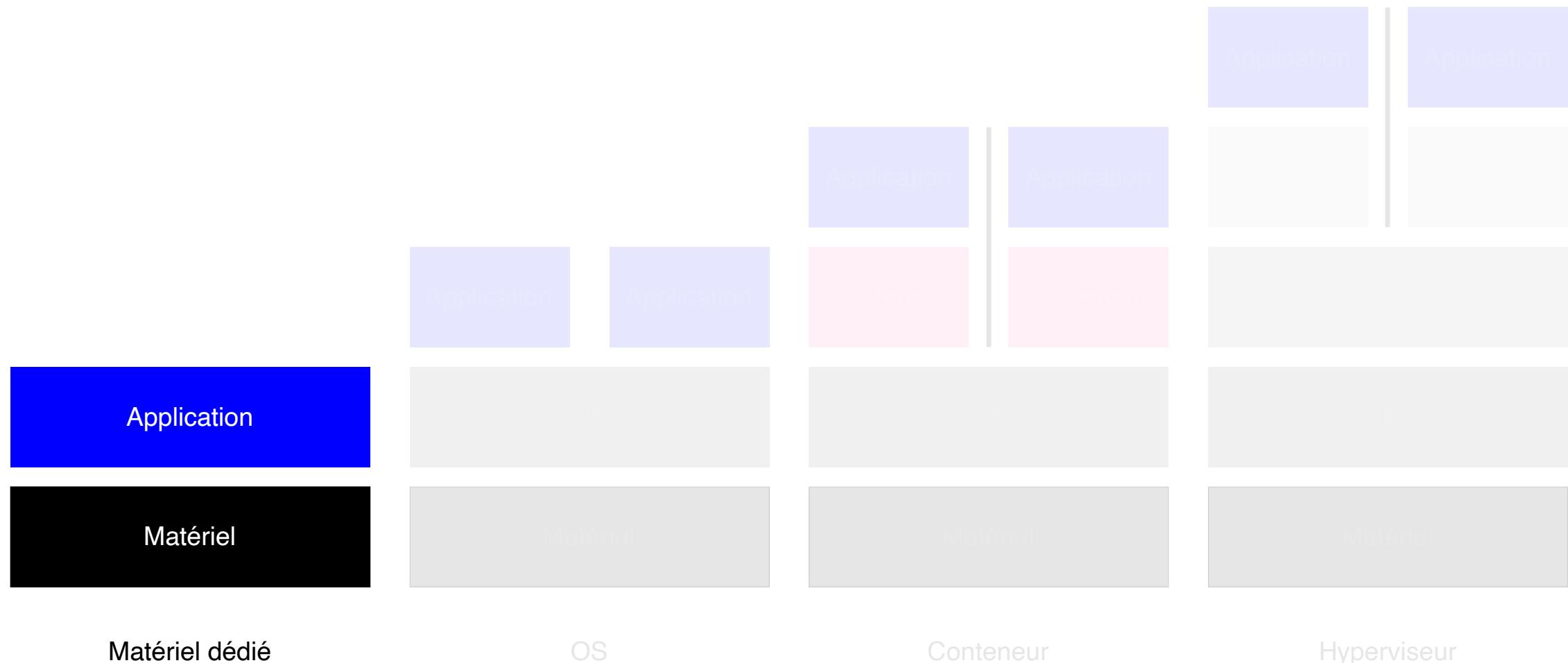
- Connaître les **techniques de virtualisation** :
 - proposées par un **OS**
 - proposées par un **hyperviseur**
 - proposées par les **conteneurs**
 - ...
- Comprendre **leurs fonctionnements, leurs limites et leurs performances relatives**.

DÉMARCHE POUR ABORDER CE SUJET

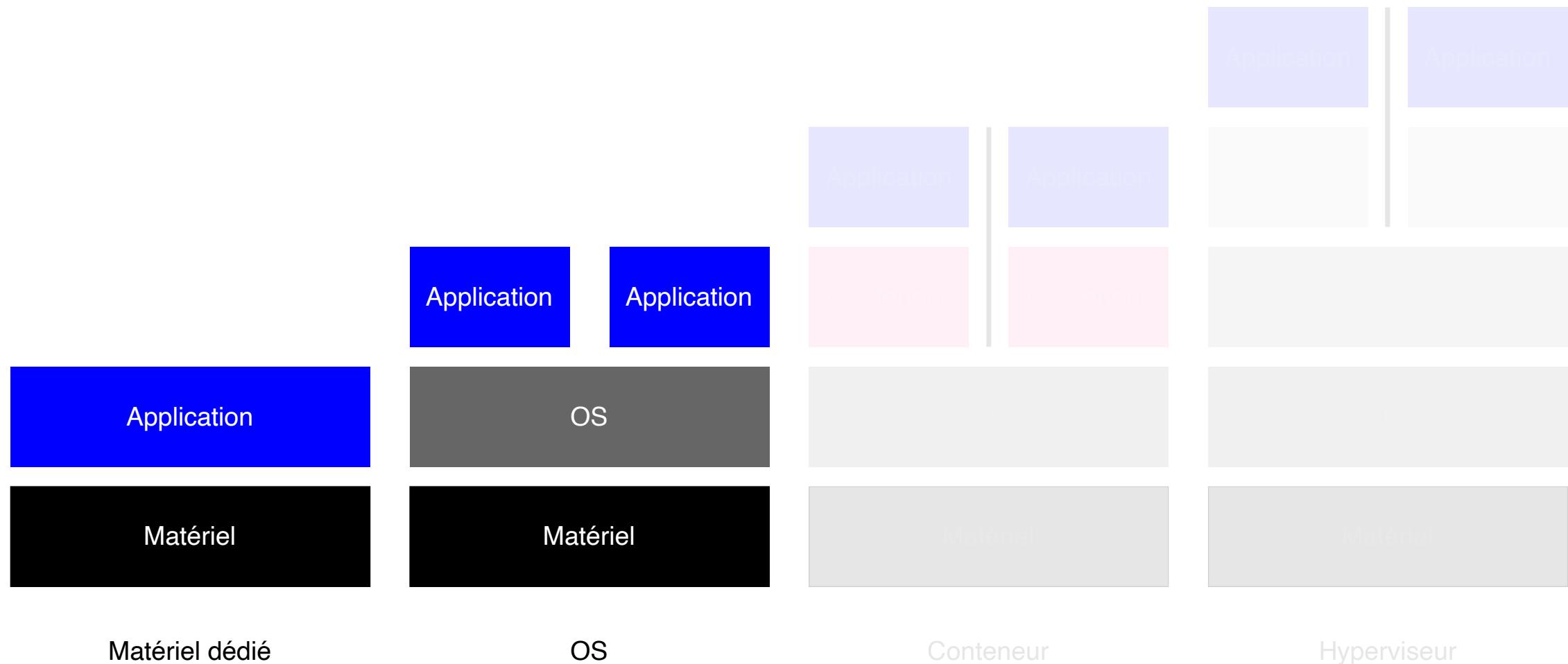
- Connaître les **techniques de virtualisation** :
 - proposées par un **OS**
 - proposées par un **hyperviseur**
 - proposées par les **conteneurs**
 - ...
- Comprendre **leurs fonctionnements, leurs limites et leurs performances relatives.**
- Savoir **choisir une technologie de virtualisation** en fonction **des usages.**

STADES DE VIRTUALISATION

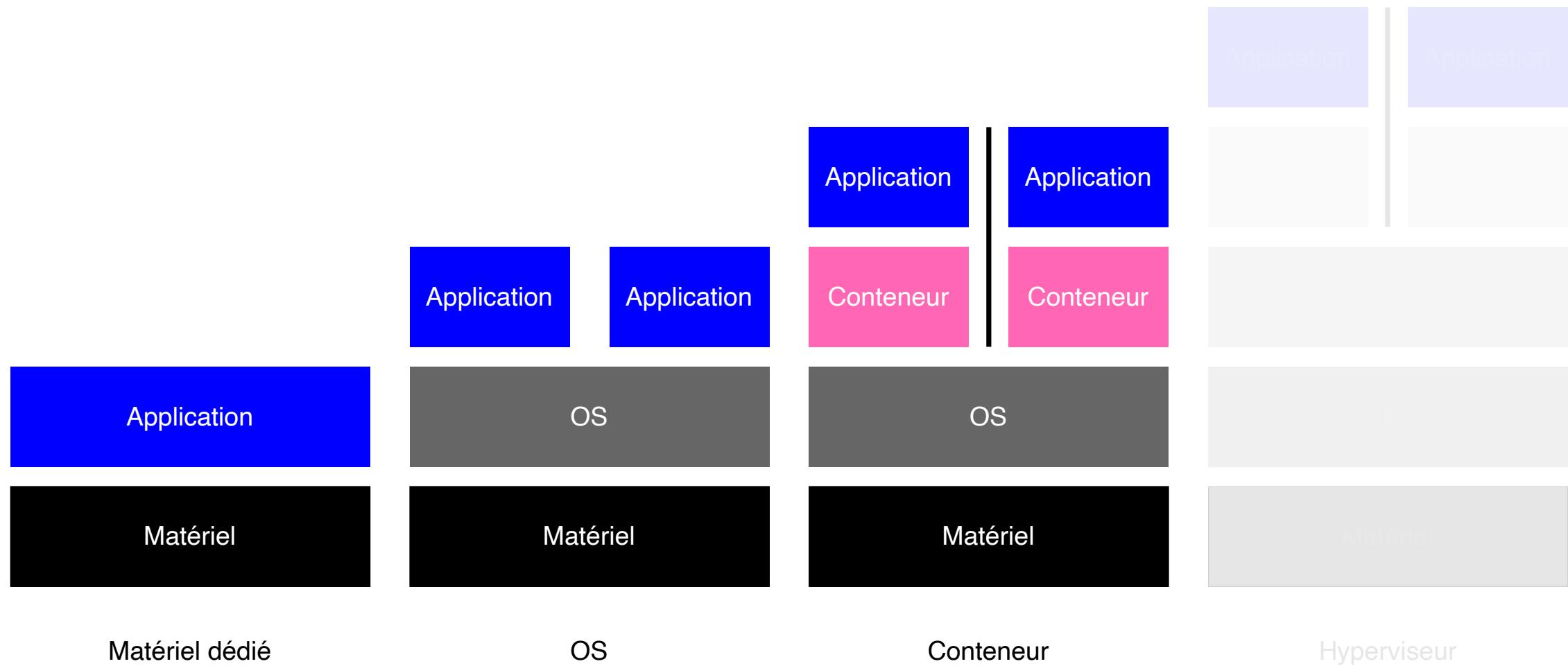
STADES DE VIRTUALISATION



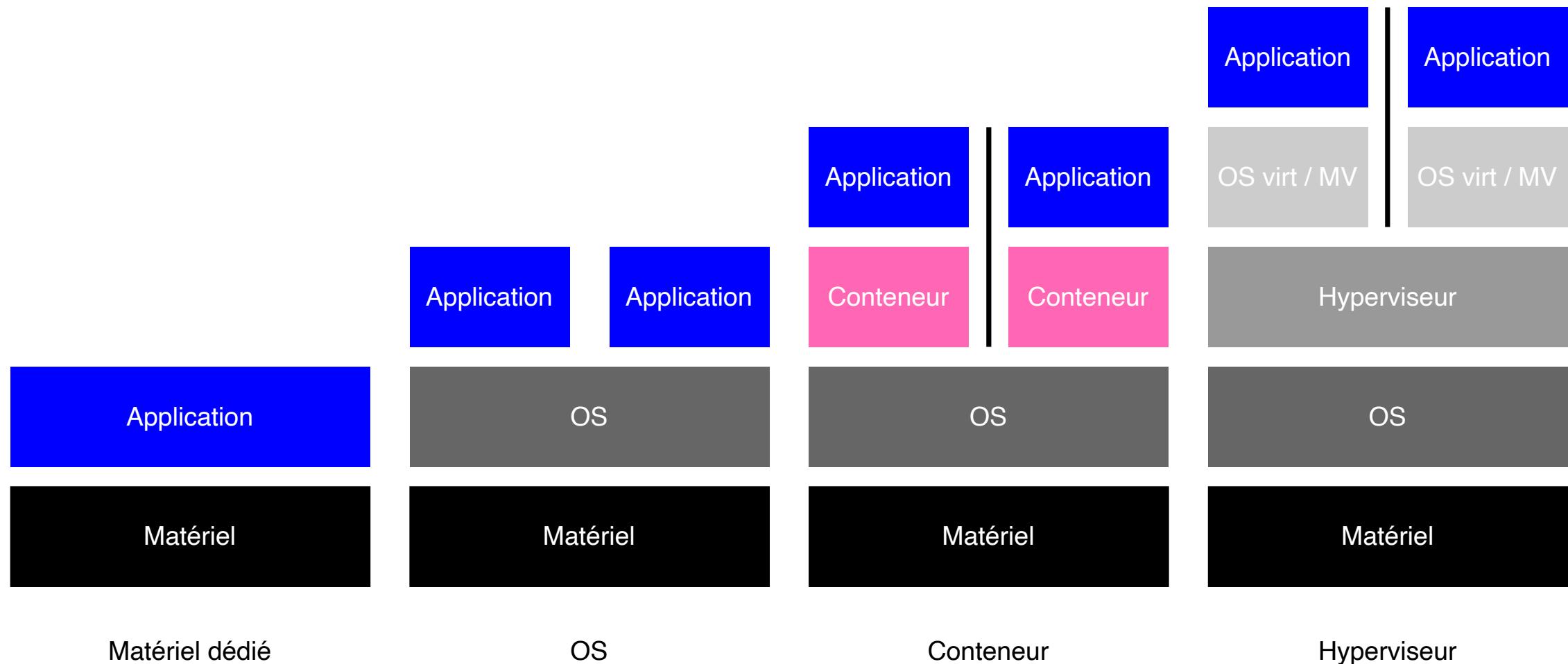
STADES DE VIRTUALISATION



STADES DE VIRTUALISATION

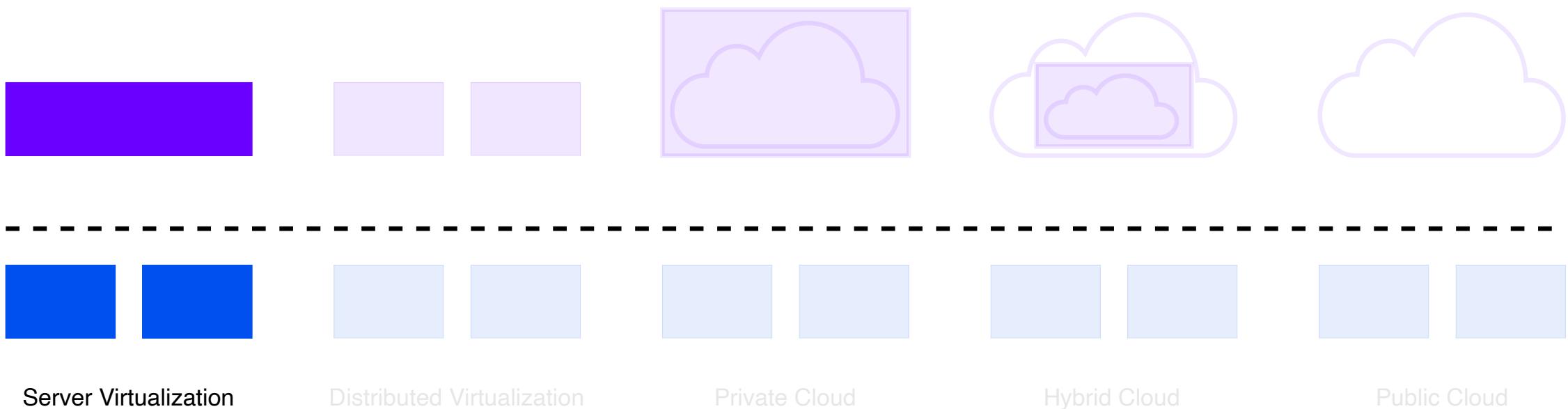


STADES DE VIRTUALISATION

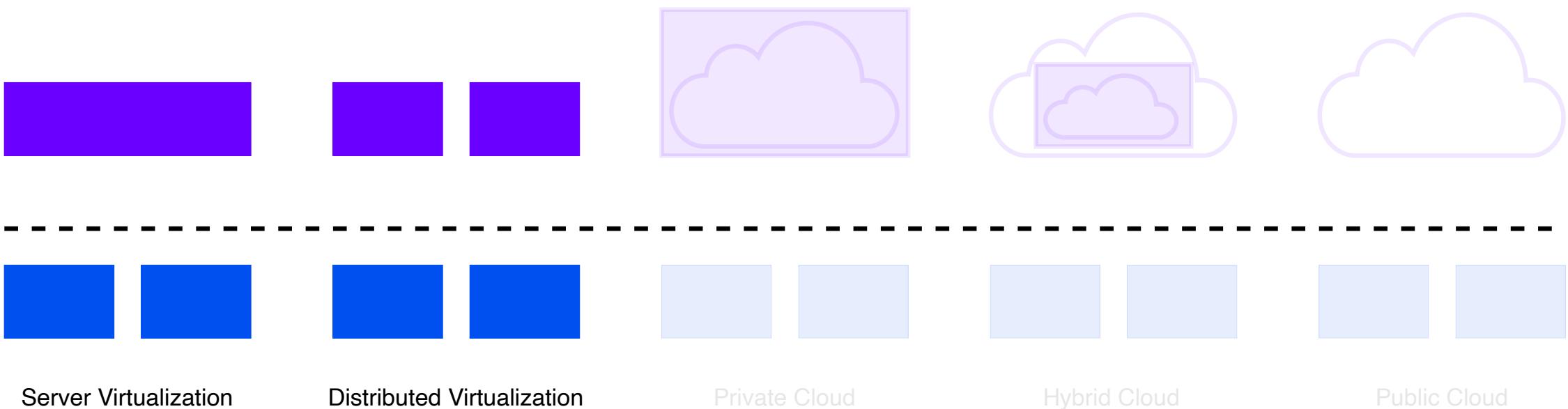


DE LA VIRTUALISATION AU CLOUD

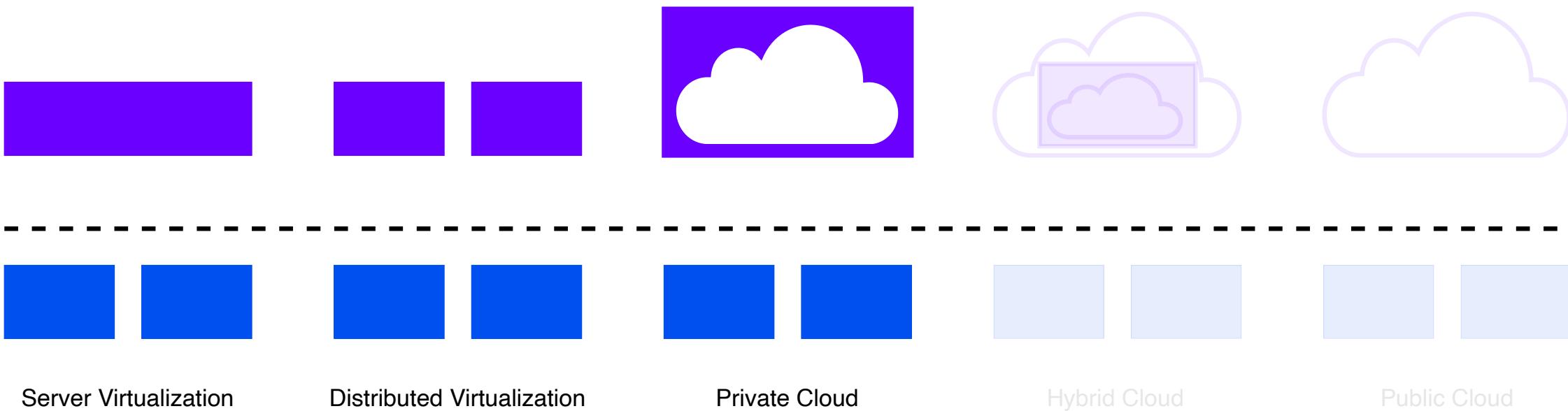
DE LA VIRTUALISATION AU CLOUD



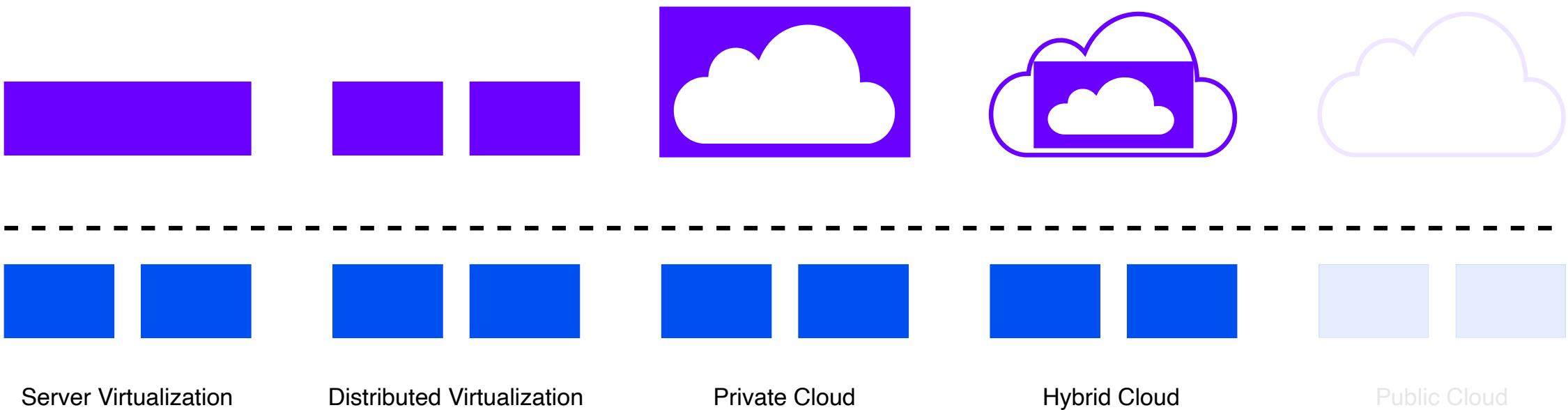
DE LA VIRTUALISATION AU CLOUD



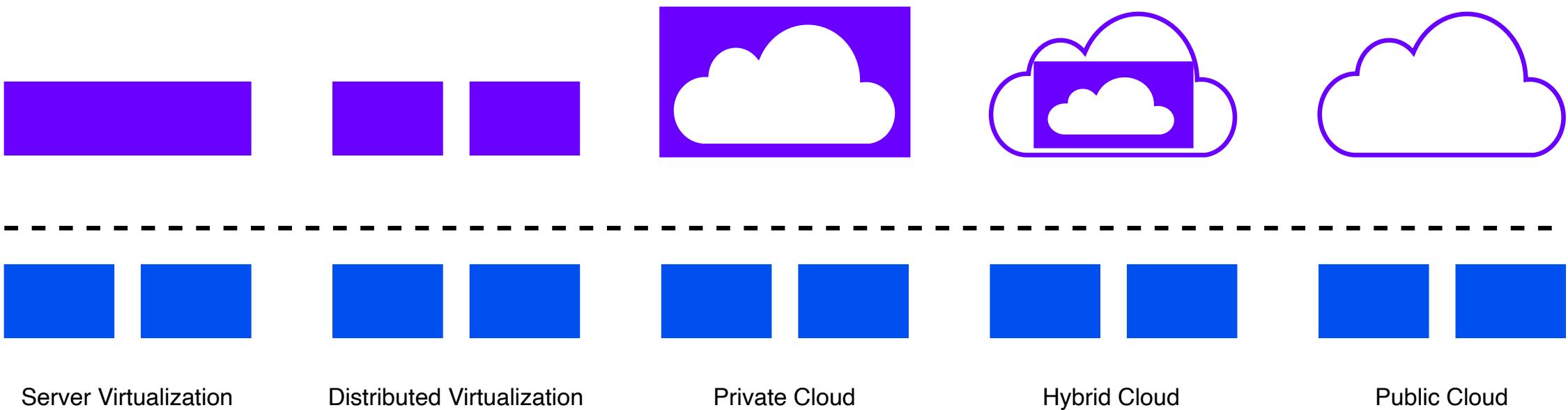
DE LA VIRTUALISATION AU CLOUD



DE LA VIRTUALISATION AU CLOUD



DE LA VIRTUALISATION AU CLOUD



CHEMINEMENT

CHEMINEMENT

- Comprendre ce sujet nécessite une compréhension

CHEMINEMENT

- Comprendre ce sujet nécessite une compréhension
 - **du matériel** : CPU, mémoire, stockage, périphériques

CHEMINEMENT

- Comprendre ce sujet nécessite une compréhension
 - **du matériel** : CPU, mémoire, stockage, périphériques
 - **des OS** : gestion de processus, gestion mémoire, sécurité

CHEMINEMENT

- Comprendre ce sujet nécessite une compréhension
 - **du matériel** : CPU, mémoire, stockage, périphériques
 - **des OS** : gestion de processus, gestion mémoire, sécurité
 - et **du réseau** : commutation, routage, encapsulation (non abordé)

CHEMINEMENT

- Comprendre ce sujet nécessite une compréhension
 - **du matériel** : CPU, mémoire, stockage, périphériques
 - **des OS** : gestion de processus, gestion mémoire, sécurité
 - et **du réseau** : commutation, routage, encapsulation (non abordé)
- Comprendre ces sujets est aussi requis pour appréhender le sujet de **la sécurité des systèmes d'information (SSI)**

SOURCES DE CES TRANSPARENTS

Transparents reposant sur de nombreux auteurs :

*“Idir Ait-Sadoune, Christophe Jacquet, Thibault Le Meur,
Gianluca Quercini, Nicolas Sabouret, Marc-Antoine Weisser, ...”*

PLAN

- Architecture des ordinateurs
- Structure et évolution des OS
- Fonctionnalités principales d'un OS
- Virtualisation

[Retour au plan](#) - [Retour à l'accueil](#)

PLAN

- Architecture des ordinateurs
- Structure et évolution des OS
- Fonctionnalités principales d'un OS
- Virtualisation

[Retour au plan](#) - [Retour à l'accueil](#)

L'INFORMATIQUE



L'INFORMATIQUE

- L'informatique est la science du **traitement automatique de l'information**.



L'INFORMATIQUE

- L'informatique est la science du **traitement automatique de l'information**.
- Le traitement automatique de l'information s'effectue avec des **programmes informatiques** exécutés par des **machines**



L'INFORMATIQUE

- L'informatique est la science du **traitement automatique de l'information**.
- Le traitement automatique de l'information s'effectue avec des **programmes informatiques** exécutés par des **machines**
 - **les programmes (software)** décrivent le traitement à réaliser,



L'INFORMATIQUE

- L'informatique est la science du **traitement automatique de l'information**.
- Le traitement automatique de l'information s'effectue avec des **programmes informatiques** exécutés par des **machines**
 - **les programmes (software)** décrivent le traitement à réaliser,
 - **les machines (hardware)** exécutent **les programmes**.



LA NOTION D'ORDINATEUR



LA NOTION D'ORDINATEUR



- L'ordinateur désigne **un équipement informatique** permettant de traiter des informations en **exécutant des instructions**.

LA NOTION D'ORDINATEUR



- L'ordinateur désigne **un équipement informatique** permettant de traiter des informations en **exécutant des instructions**.
👉 On lui donne **des instructions** (programme/logiciel)

LA NOTION D'ORDINATEUR



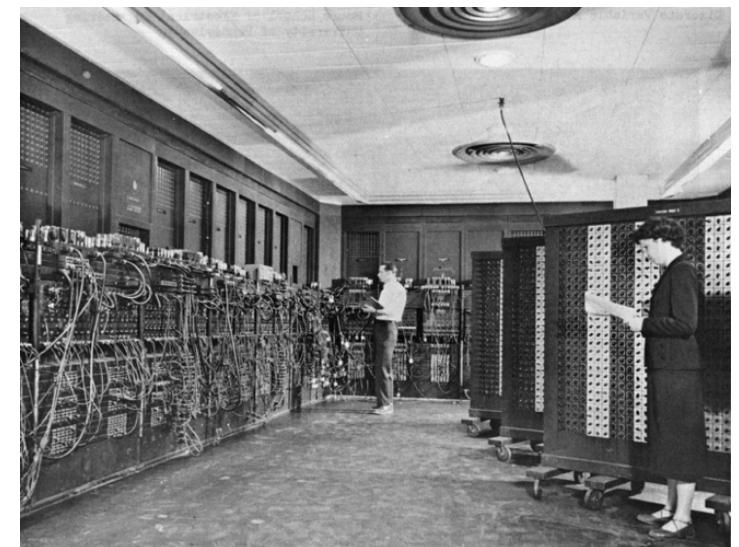
- L'ordinateur désigne **un équipement informatique** permettant de traiter des informations en **exécutant des instructions**.
 - 👉 On lui donne **des instructions** (programme/logiciel)
 - 👉 On lui donne **des données** (information)

LA NOTION D'ORDINATEUR



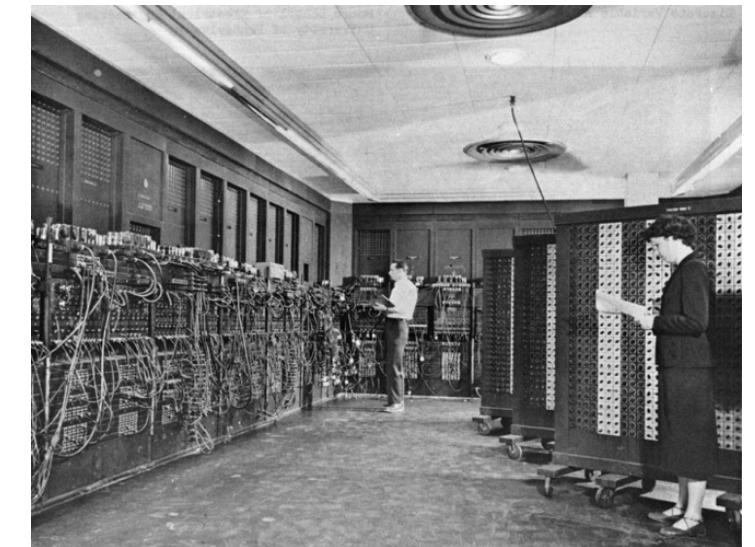
- L'ordinateur désigne **un équipement informatique** permettant de traiter des informations en **exécutant des instructions**.
 - 👉 On lui donne **des instructions** (programme/logiciel)
 - 👉 On lui donne **des données** (information)
 - 👉 Il **transforme** les données

ENIAC - 1946



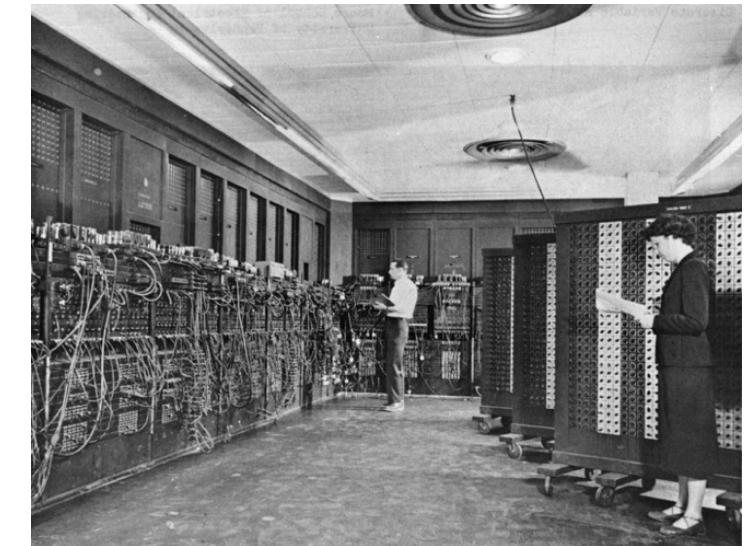
ENIAC - 1946

- Construit de 1943 à 1946 par John Mauchley et John Eckert à l'université de Pennsylvanie



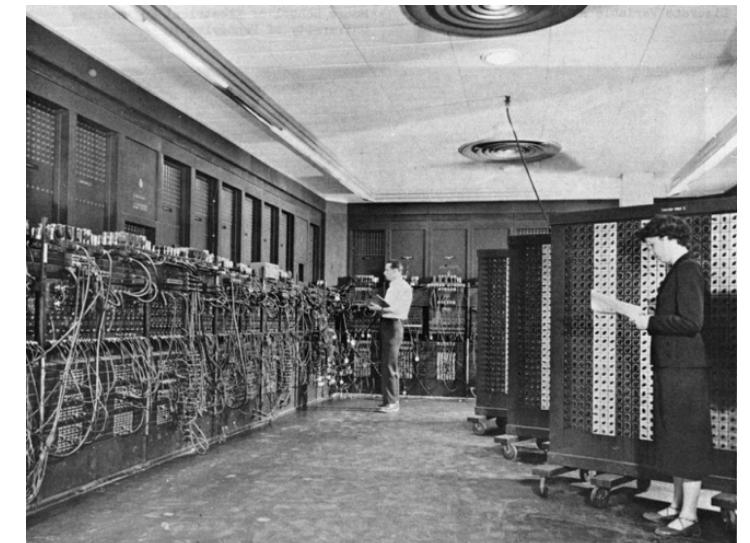
ENIAC - 1946

- Construit de 1943 à 1946 par John Mauchley et John Eckert à l'université de Pennsylvanie
- Premier ordinateur entièrement électronique (utilise des tubes à vide).



ENIAC - 1946

- Construit de 1943 à 1946 par John Mauchley et John Eckert à l'université de Pennsylvanie
- Premier ordinateur **entièrement électronique** (utilise des **tubes à vide**).
- Programmé pour résoudre tous **les problèmes calculatoires**.



HP 3000 - 1972



HP 3000 - 1972

- Le **mini-ordinateur** a été une innovation des années 1970.



HP 3000 - 1972

- Le **mini-ordinateur** a été une innovation des années **1970**.
- L'intégration de **circuits intégrés à grande échelle** conduit au développement des **micro-processeurs**.

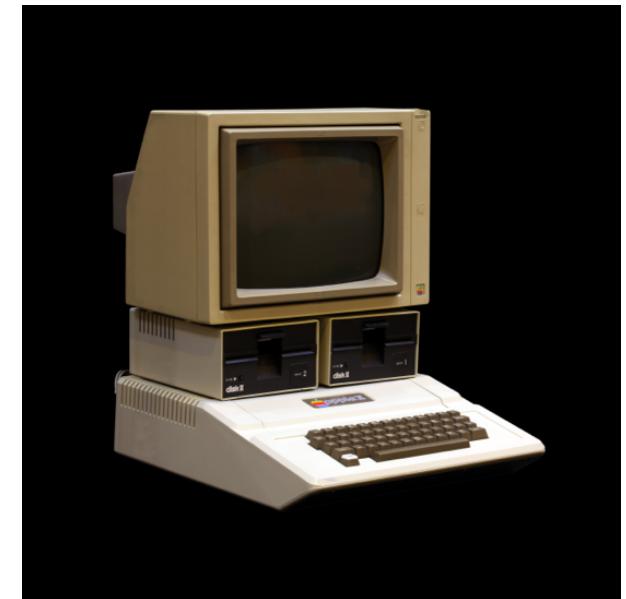


APPLE II - 1977



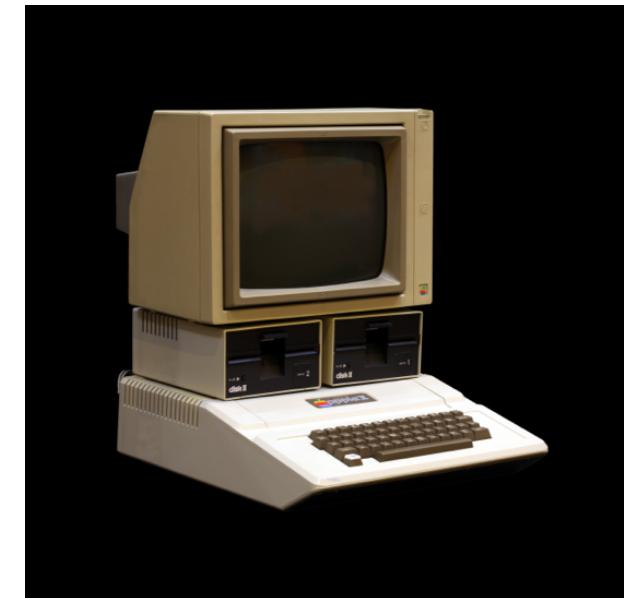
APPLE II - 1977

- Un des premiers **ordinateurs personnels**
à **micro-processeur** fabriqué à grande échelle

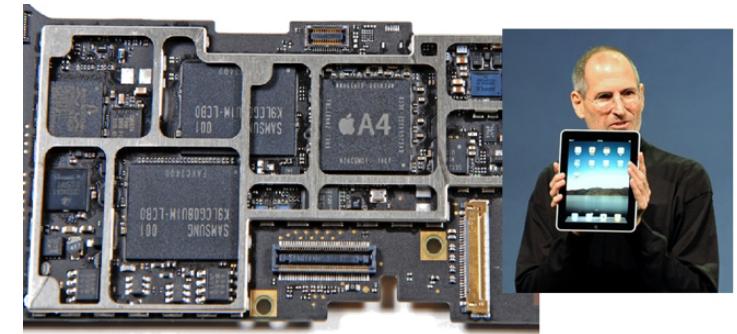


APPLE II - 1977

- Un des premiers **ordinateurs personnels** à **micro-processeur** fabriqué à grande échelle
- Conçu par **Steve Wozniak**, commercialisé le **10 juin 1977** par **Apple**

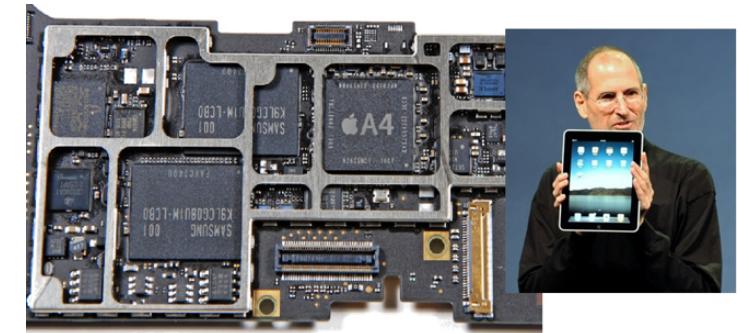


LES ORDINATEURS D'AUJOURD'HUI



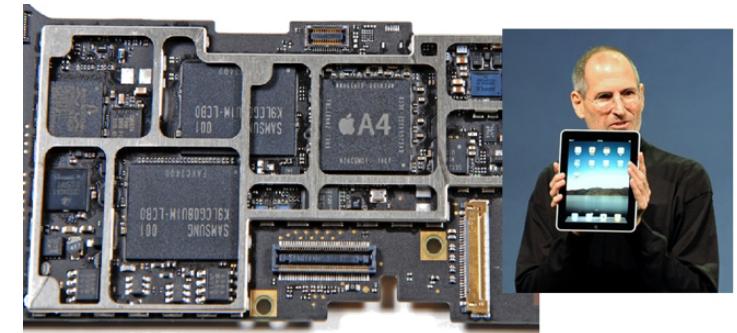
LES ORDINATEURS D'AUJOURD'HUI

- System on a Chip (**SOC**) : un système complet embarqué dans une puce (**circuit intégré**).



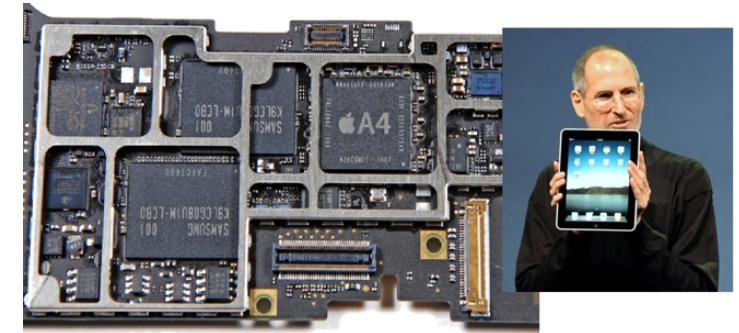
LES ORDINATEURS D'AUJOURD'HUI

- System on a Chip (**SOC**) : un système complet embarqué dans une puce (**circuit intégré**).
- Un **circuit intégré** peut comprendre :
 - un ou plusieurs microprocesseurs
 - de la mémoire
 - des périphériques d'interface
 - ou tout autre composant

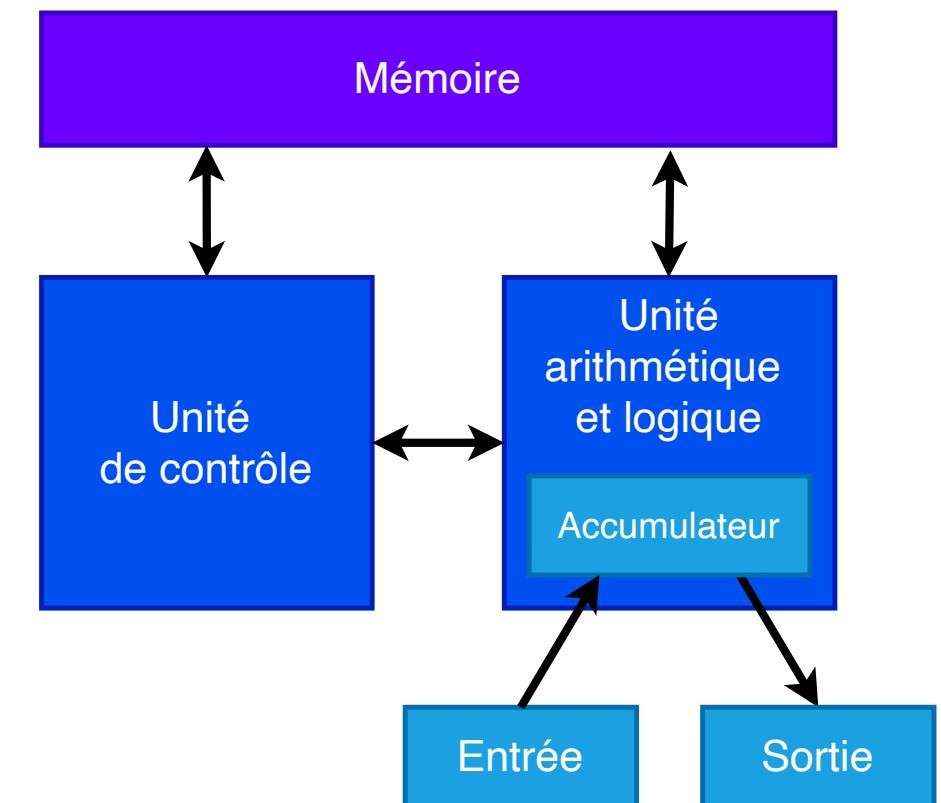


LES ORDINATEURS D'AUJOURD'HUI

- System on a Chip (**SOC**) : un système complet embarqué dans une puce (**circuit intégré**).
- Un **circuit intégré** peut comprendre :
 - un ou plusieurs microprocesseurs
 - de la mémoire
 - des périphériques d'interface
 - ou tout autre composant

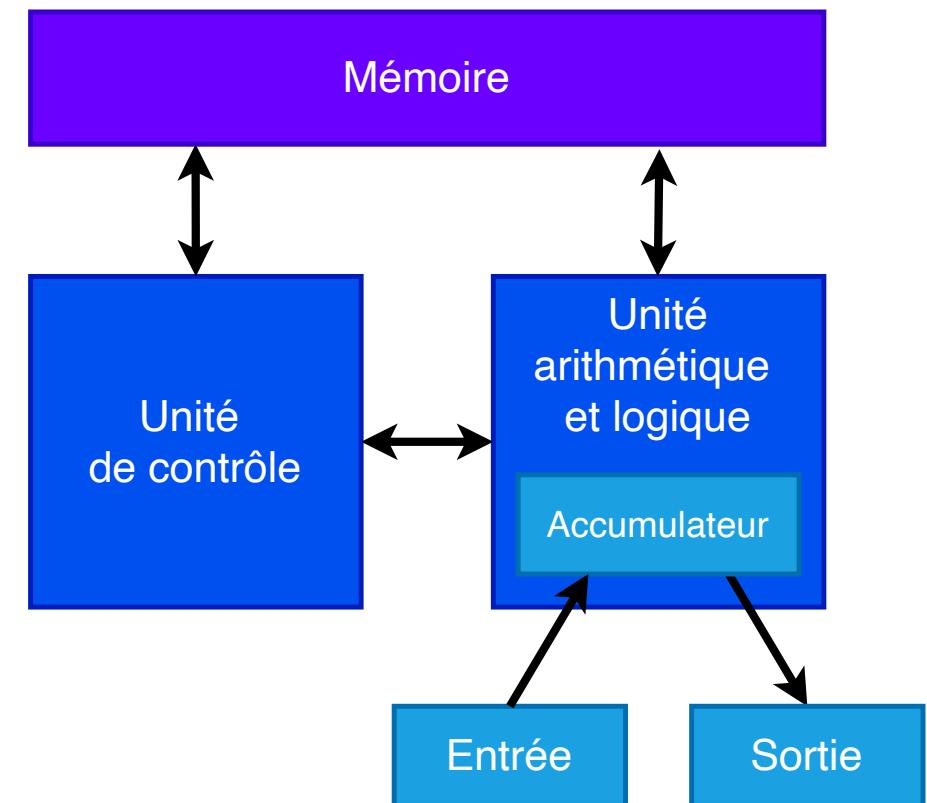


L'ARCHITECTURE DE VON NEUMANN

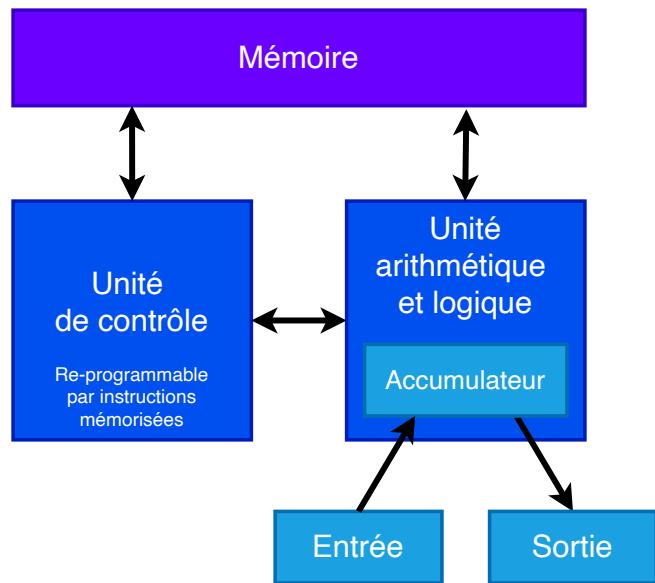


L'ARCHITECTURE DE VON NEUMANN

- L'architecture de von Neumann : un **modèle** pour un ordinateur avec une **mémoire unique** pour conserver
 - les instructions
 - et les données



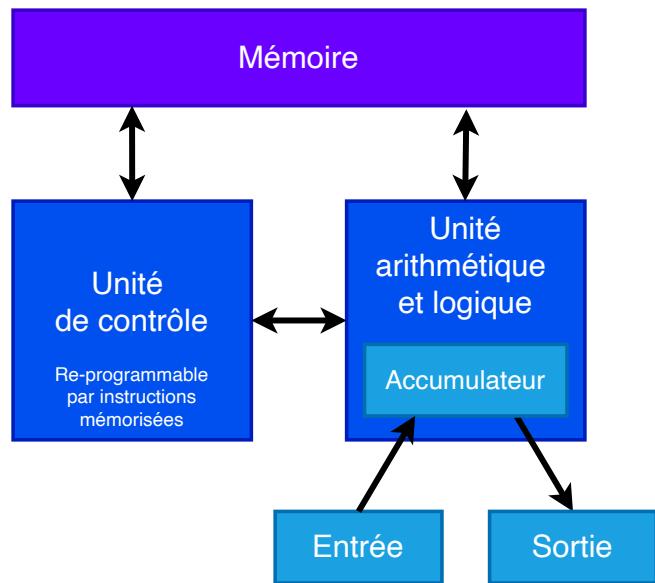
VON NEUMANN / L'ORDINATEUR MODERNE



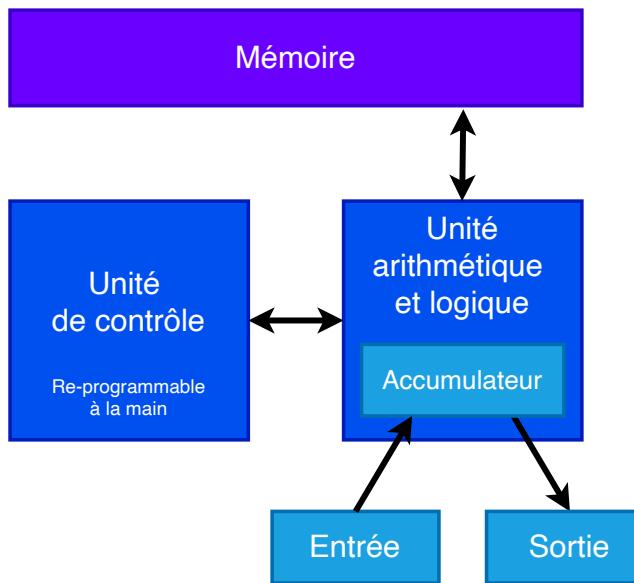
Von Neumann

source : [la thèse d'Alexandre Brunet](#)

VON NEUMANN / L'ORDINATEUR MODERNE



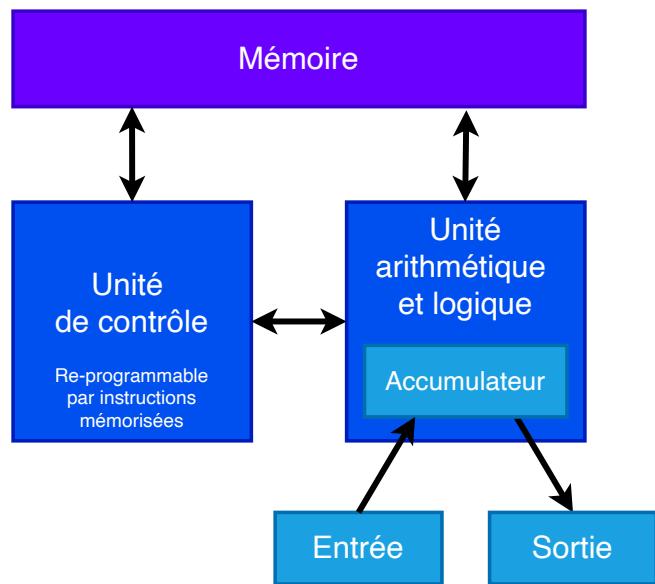
Von Neumann



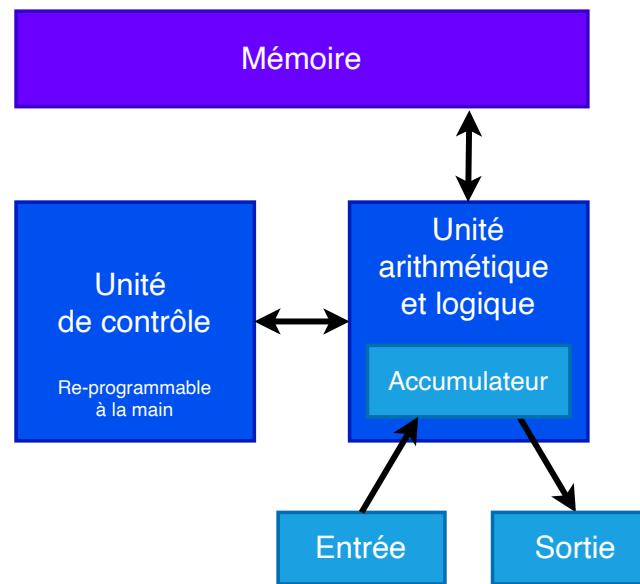
ENIAC

source : [la thèse d'Alexandre Brunet](#)

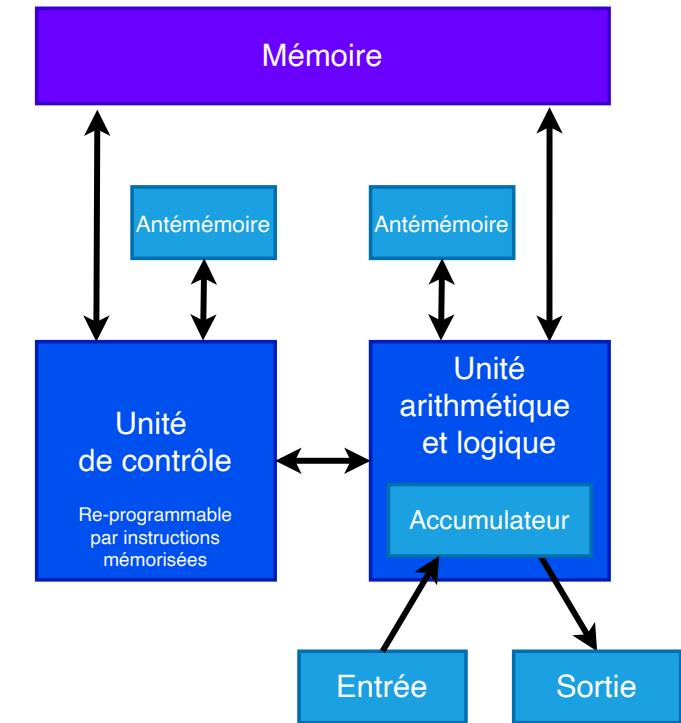
VON NEUMANN / L'ORDINATEUR MODERNE



Von Neumann



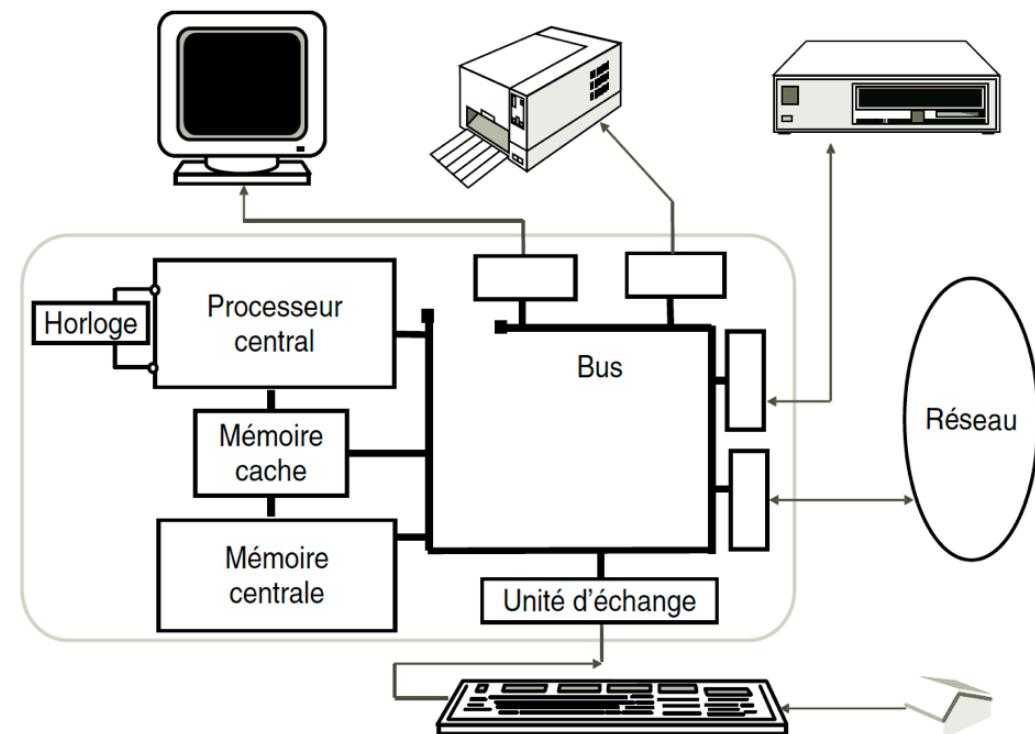
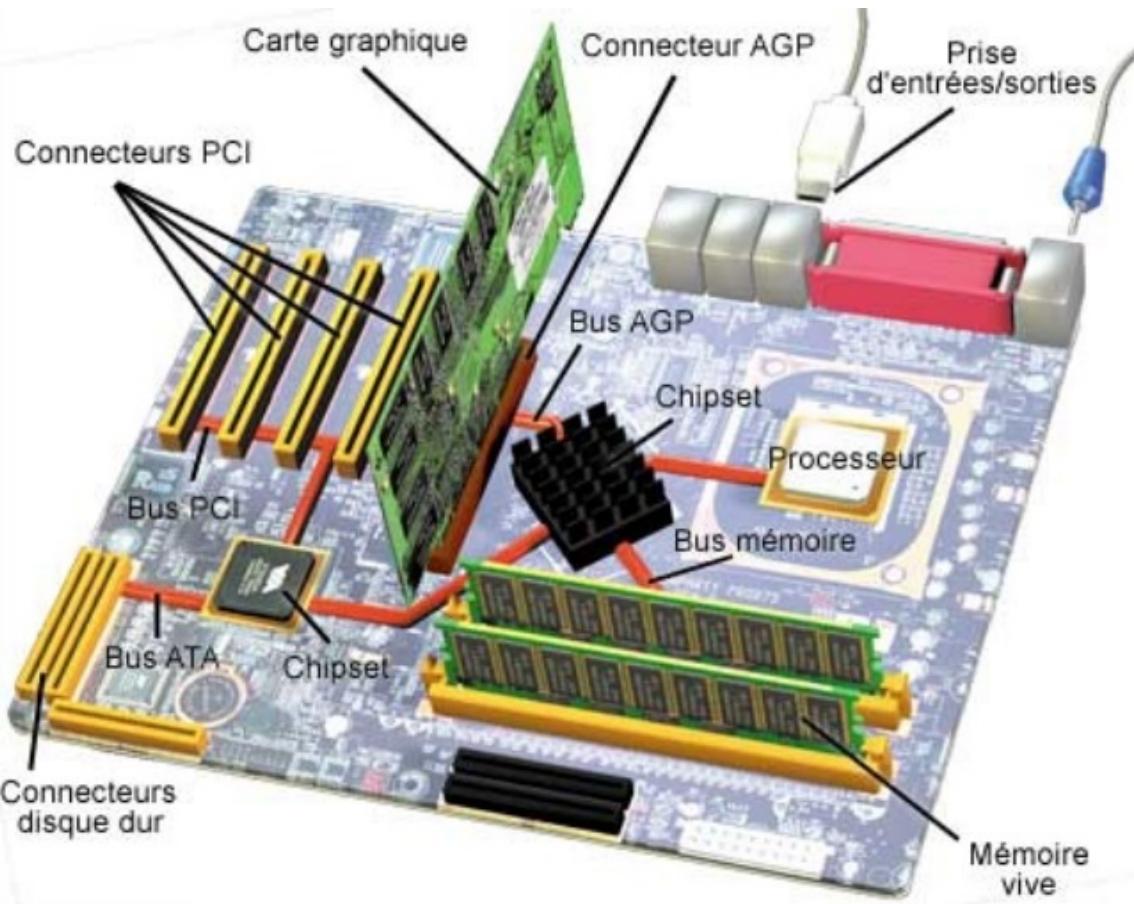
ENIAC



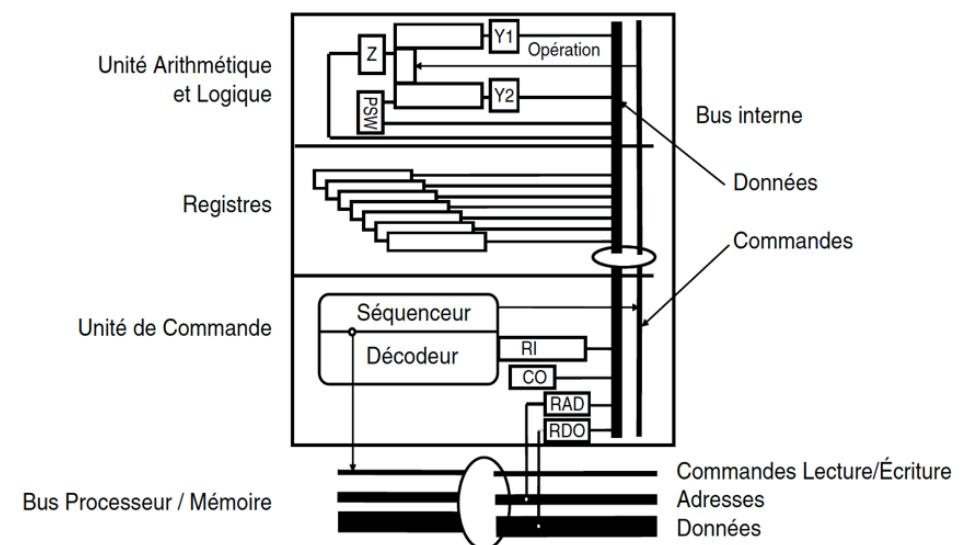
Ordinateur personnel

source : [la thèse d'Alexandre Brunet](#)

STRUCTURE GÉNÉRALE D'UN ORDINATEUR

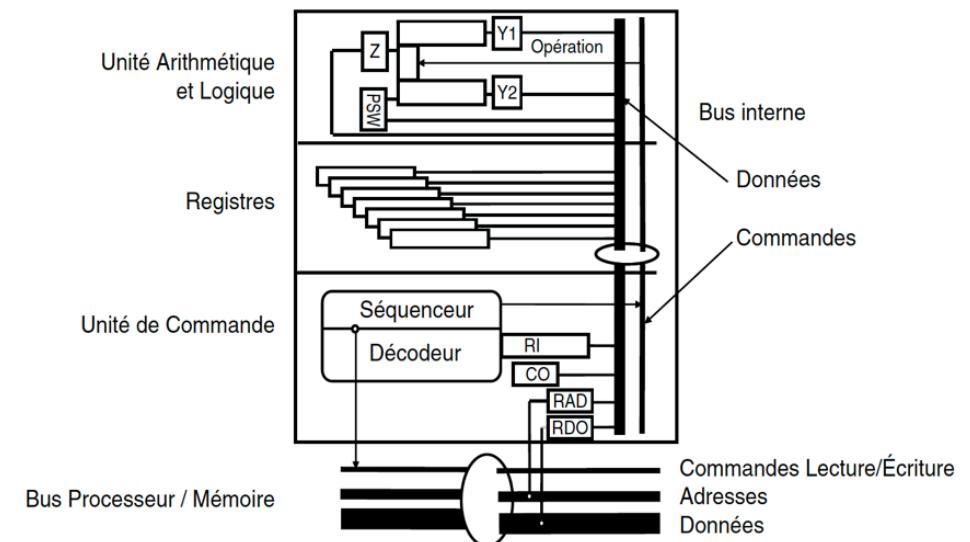


L'ARCHITECTURE D'UN MICROPROCESSEUR



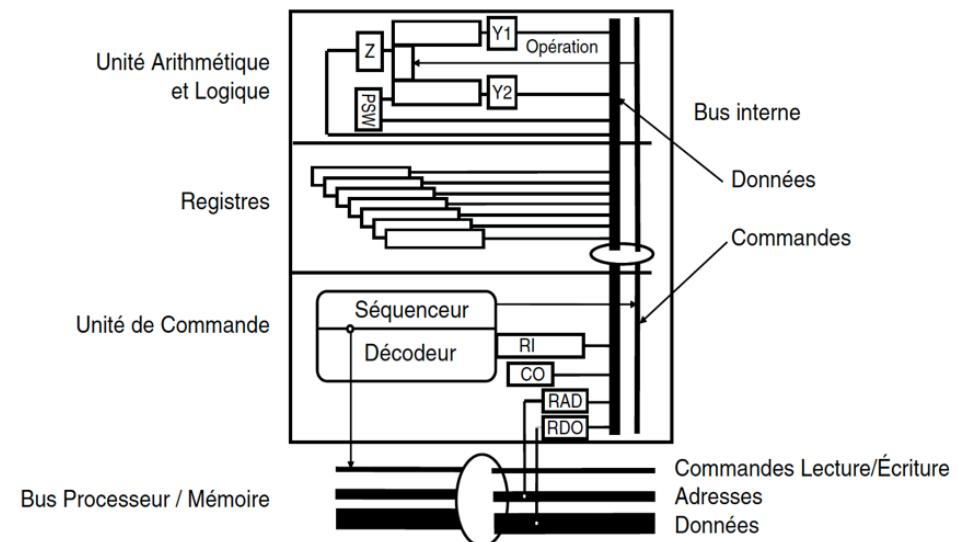
L'ARCHITECTURE D'UN MICROPROCESSEUR

- Le microprocesseur (**CPU**) exécute les *instructions machines* placées en mémoire centrale.



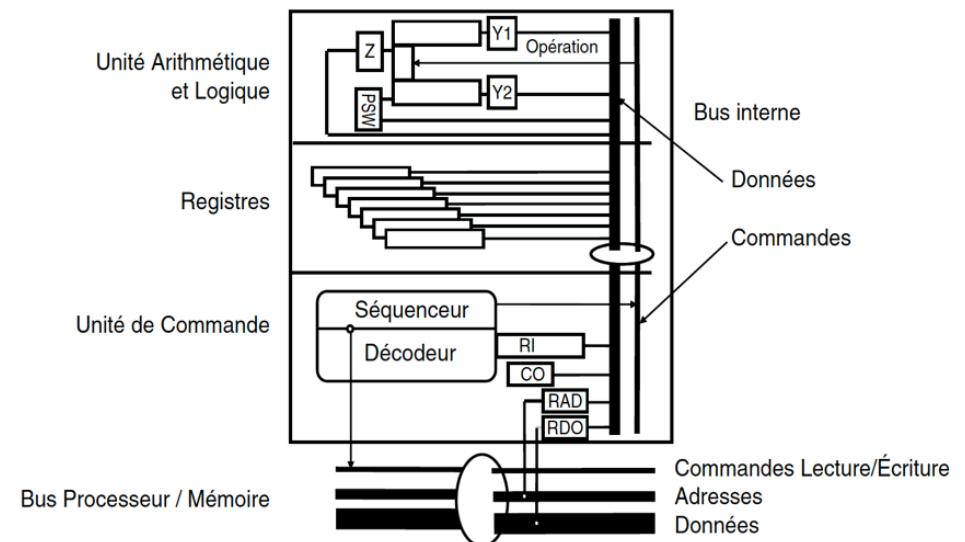
L'ARCHITECTURE D'UN MICROPROCESSEUR

- Le microprocesseur (**CPU**) exécute les *instructions machines* placées en mémoire centrale.
- Le **CPU** est constitué de quatre parties



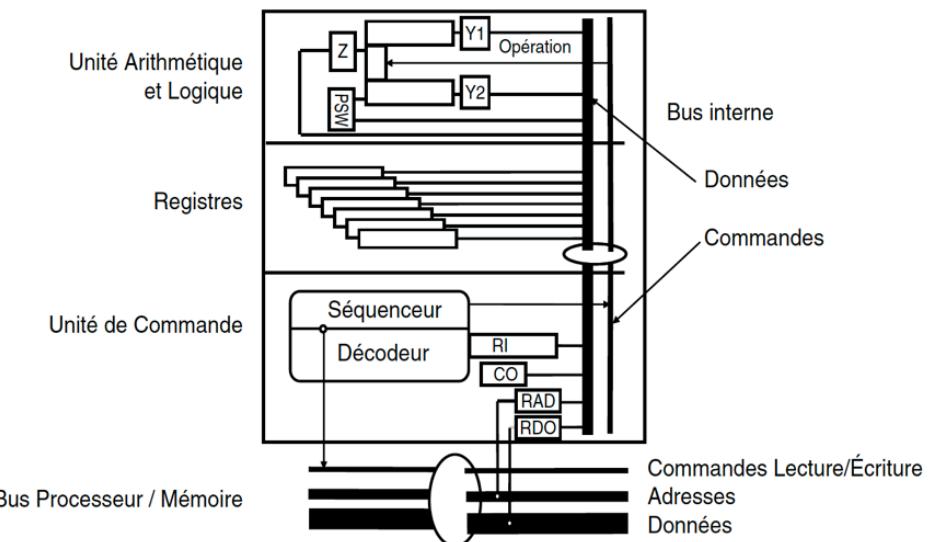
L'ARCHITECTURE D'UN MICROPROCESSEUR

- Le microprocesseur (**CPU**) exécute les *instructions machines* placées en mémoire centrale.
- Le **CPU** est constitué de quatre parties
 1. l'unité arithmétique et logique (**UAL**),



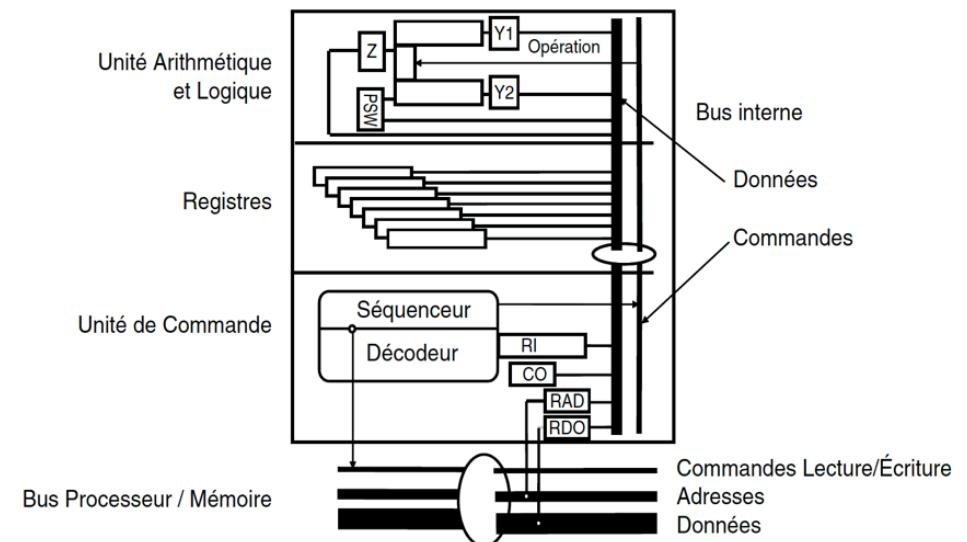
L'ARCHITECTURE D'UN MICROPROCESSEUR

- Le microprocesseur (**CPU**) exécute les *instructions machines* placées en mémoire centrale.
- Le **CPU** est constitué de quatre parties
 1. l'unité arithmétique et logique (**UAL**),
 2. les registres,



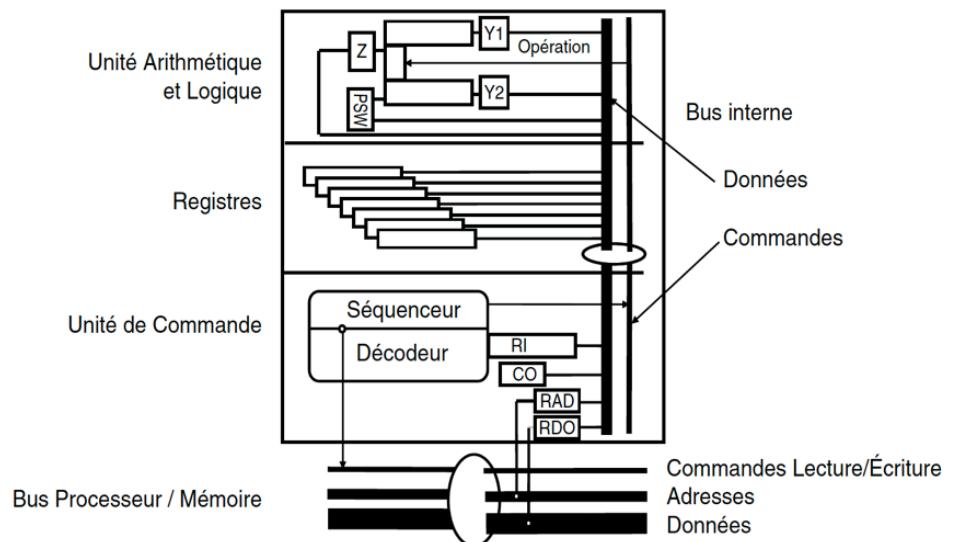
L'ARCHITECTURE D'UN MICROPROCESSEUR

- Le microprocesseur (**CPU**) exécute les *instructions machines* placées en mémoire centrale.
- Le **CPU** est constitué de quatre parties
 1. l'unité arithmétique et logique (**UAL**),
 2. les registres,
 3. l'unité de commande,



L'ARCHITECTURE D'UN MICROPROCESSEUR

- Le microprocesseur (**CPU**) exécute les *instructions machines* placées en mémoire centrale.
- Le **CPU** est constitué de quatre parties
 1. l'unité arithmétique et logique (**UAL**),
 2. les registres,
 3. l'unité de commande,
 4. le bus de communication interne.



LE MICROPROCESSEUR ET SES REGISTRES

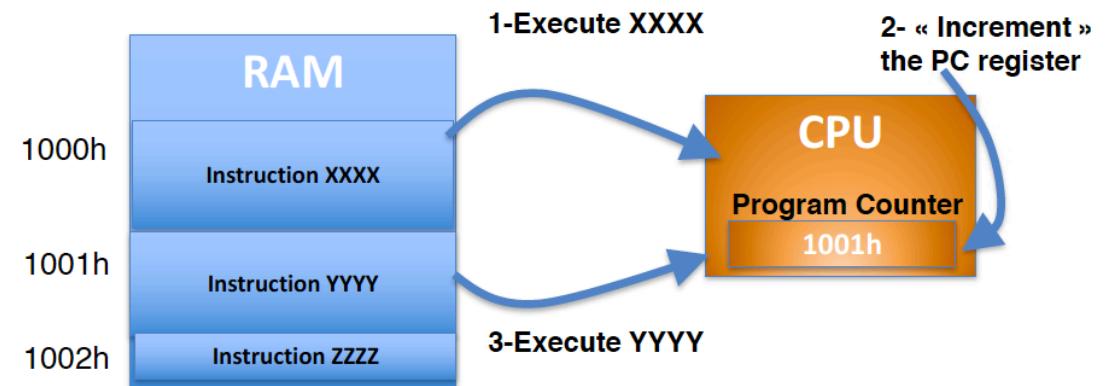
LE MICROPROCESSEUR ET SES REGISTRES

- **Registres génériques**
 - Peuvent contenir des données, adresses mémoire
 - Utilisés comme opérandes de calculs, stockage temporaire, ...

LE MICROPROCESSEUR ET SES REGISTRES

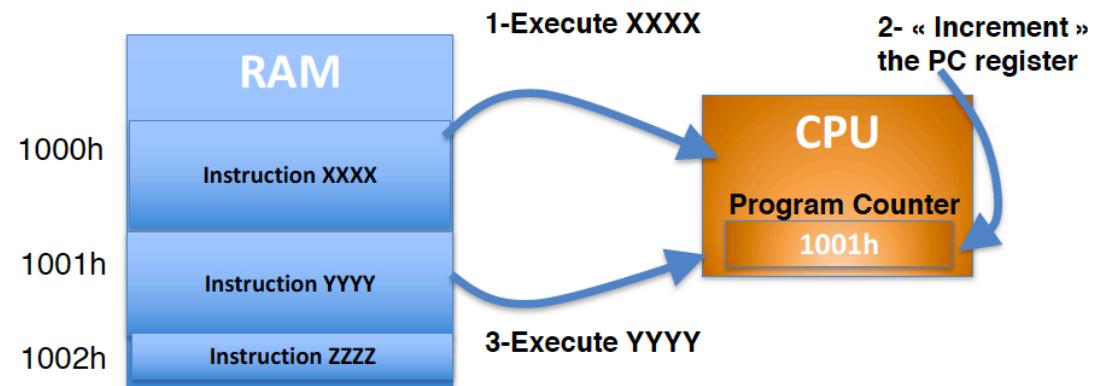
- **Registres génériques**
 - Peuvent contenir des données, adresses mémoire
 - Utilisés comme opérandes de calculs, stockage temporaire, ...
- **Registres spécialisés**, comme par exemple :
 - Program Counter
 - Stack Pointer
 - Interrupt Description Table
 - Page Table Pointer
 - ...

CHARGEMENT D'UN PROGRAMME EN MÉMOIRE



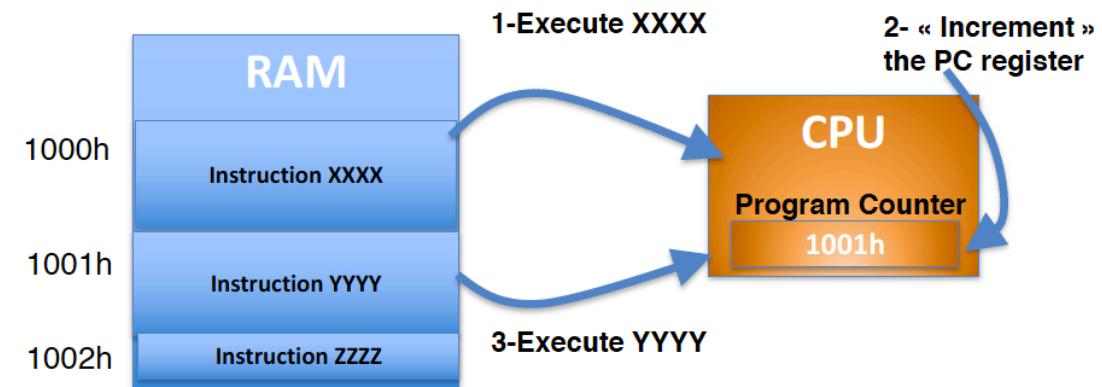
CHARGEMENT D'UN PROGRAMME EN MÉMOIRE

- Un programme est un flux d'instructions.



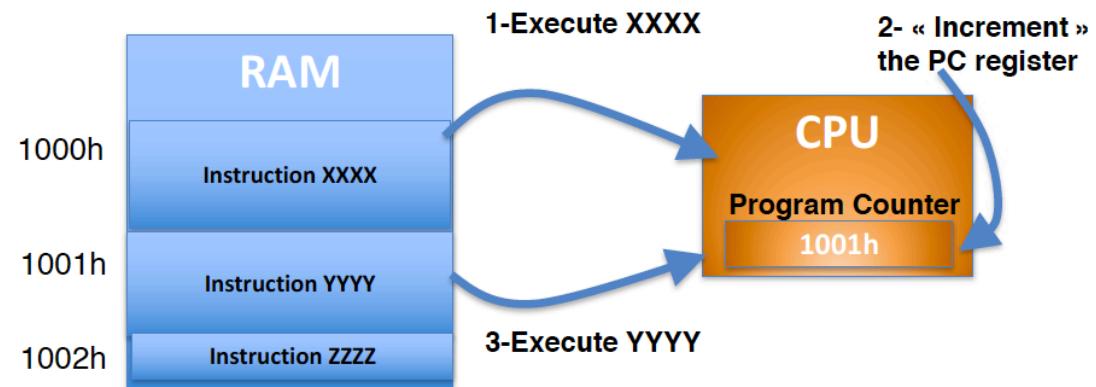
CHARGEMENT D'UN PROGRAMME EN MÉMOIRE

- Un programme est un flux d'instructions.
- Pour être exécutable, un programme doit être présent en RAM.



CHARGEMENT D'UN PROGRAMME EN MÉMOIRE

- Un programme est un flux d'instructions.
- Pour être exécutable, un programme doit être présent en RAM.
- Le registre Program Counter (PC) du CPU indique l'adresse de *la prochaine instruction à exécuter*.



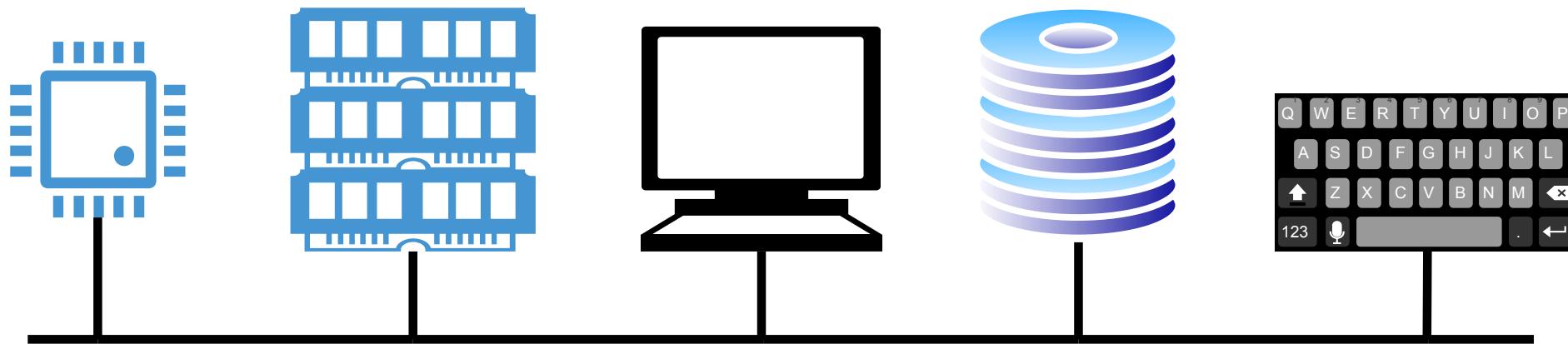
LE FONCTIONNEMENT DE L'ORDINATEUR

LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?

LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



Un calcule de base

LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



Un calcule de base

LE FONCTIONNEMENT DE L'ORDINATEUR

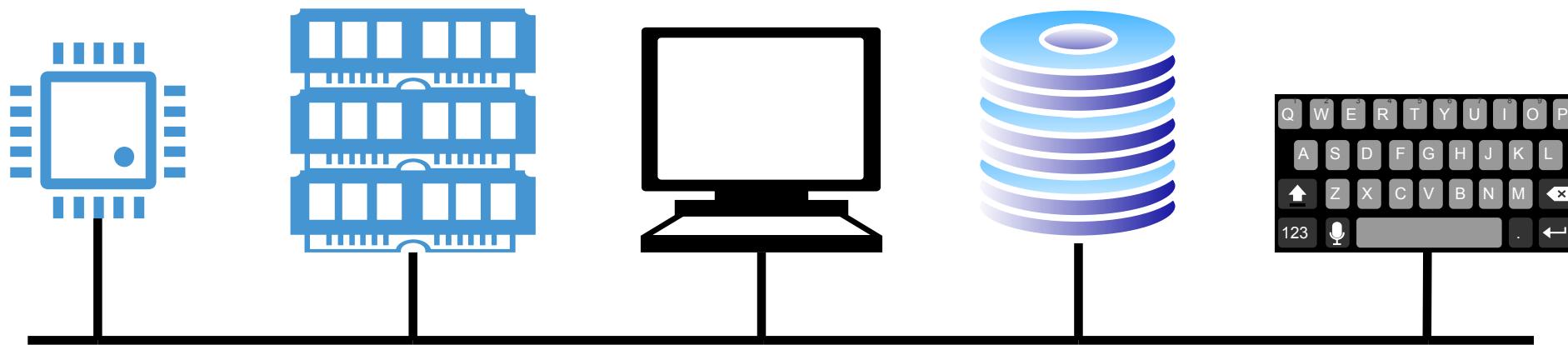
Comment fonctionne un ordinateur ?



Un calcule de base

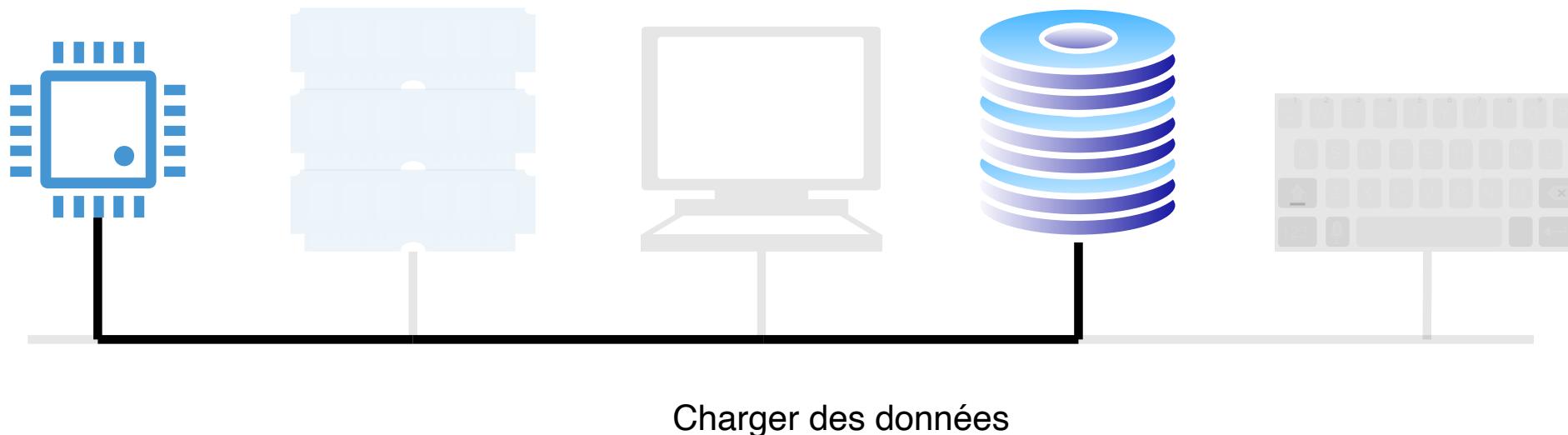
LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



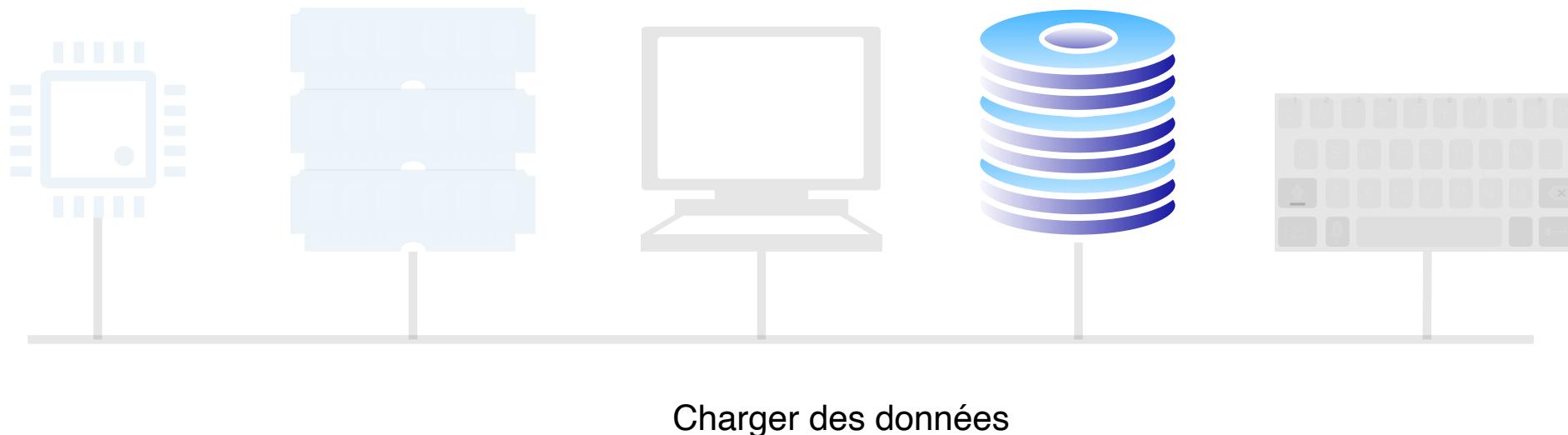
LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



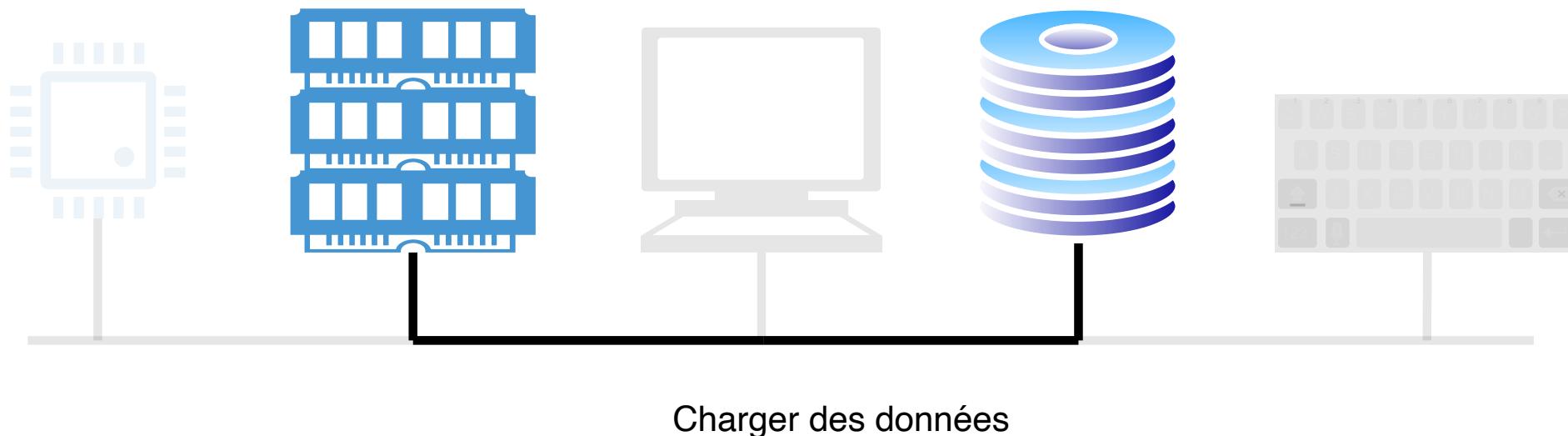
LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



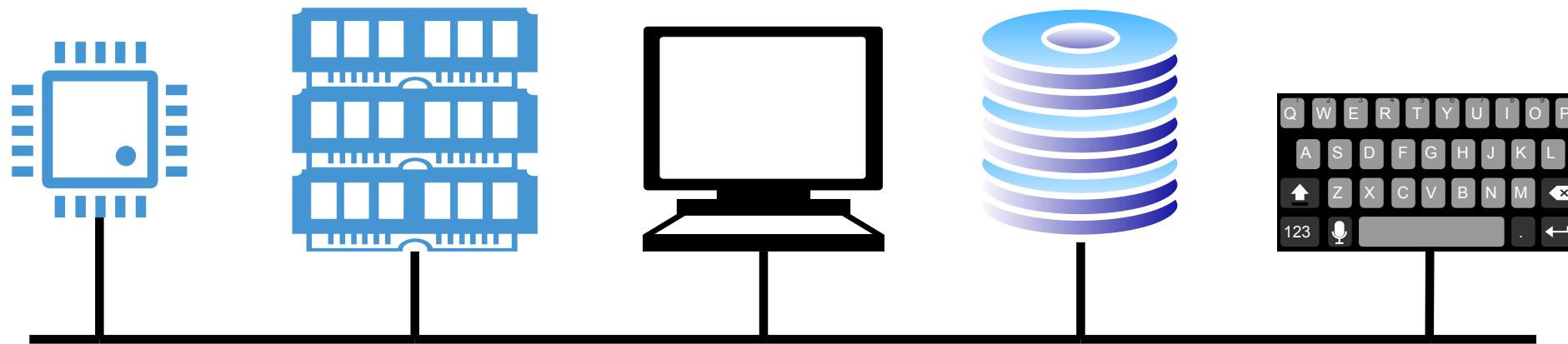
LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



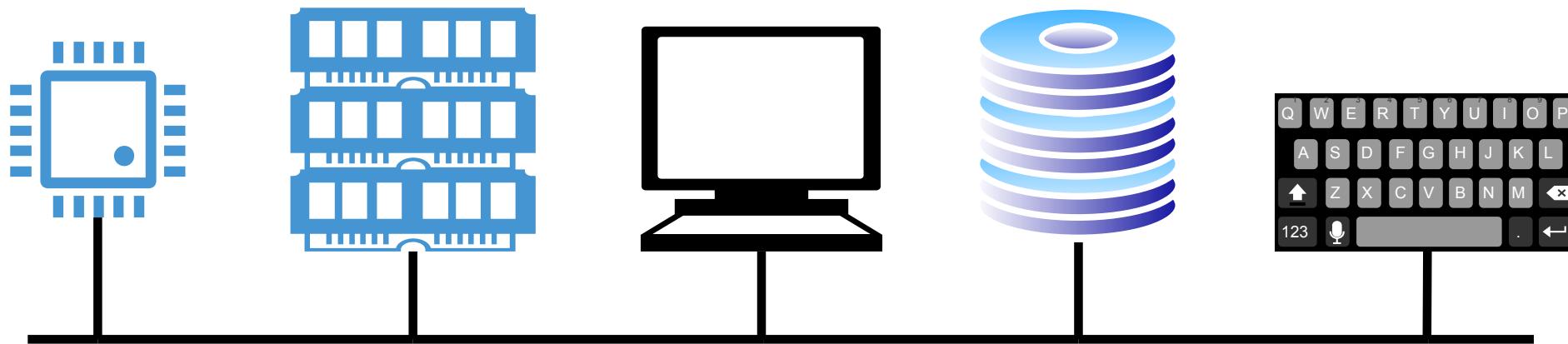
LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



LE FONCTIONNEMENT DE L'ORDINATEUR

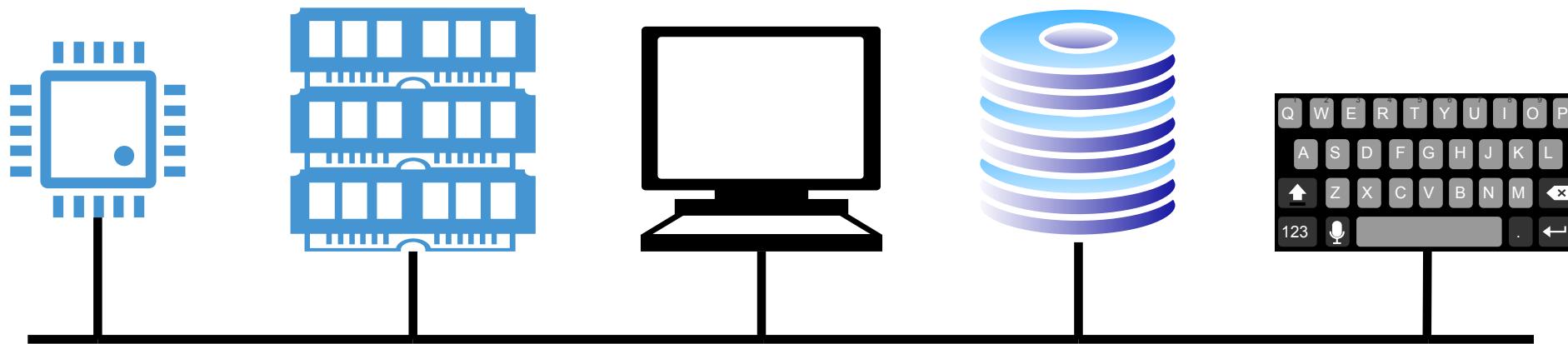
Comment fonctionne un ordinateur ?



Tout cela n'est que des fils électriques ...

LE FONCTIONNEMENT DE L'ORDINATEUR

Comment fonctionne un ordinateur ?



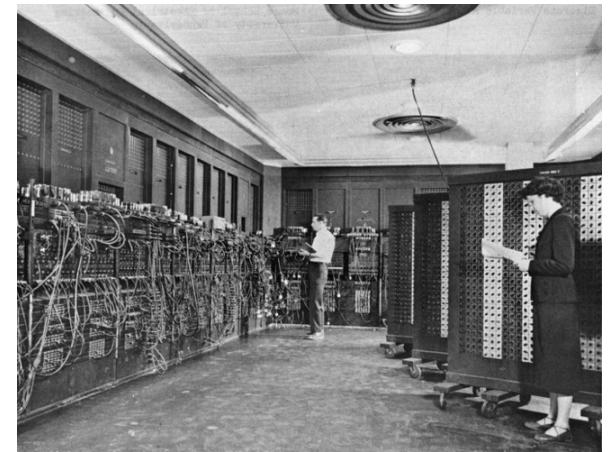
Tout cela n'est que des fils électriques ...
... qu'on allume et qu'on éteint.

PLAN

- Architecture des ordinateurs
- Structure et évolution des OS
- Fonctionnalités principales d'un OS
- Virtualisation

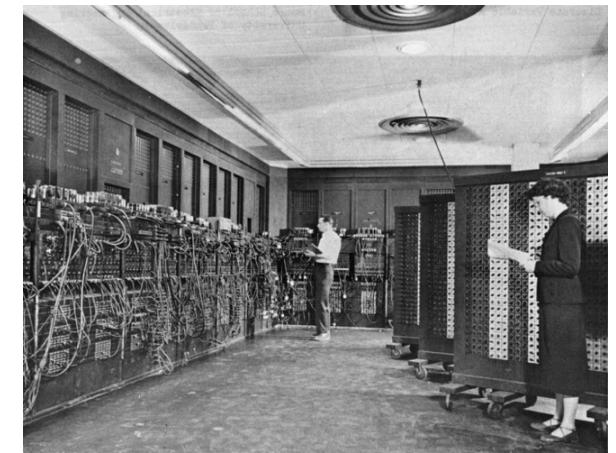
[Retour au plan](#) - [Retour à l'accueil](#)

AUTREFOIS : ENIAC



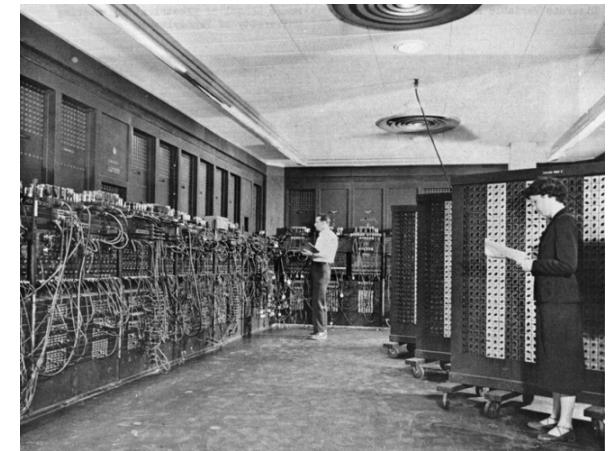
AUTREFOIS : ENIAC

- Premier ordinateur **entièlement électronique** :
 - 18 000 tubes à vide
 - 1 500 relais
 - 20 registres de 10 chiffres décimaux
 - programmé à l'aide de 6 000 commutateurs



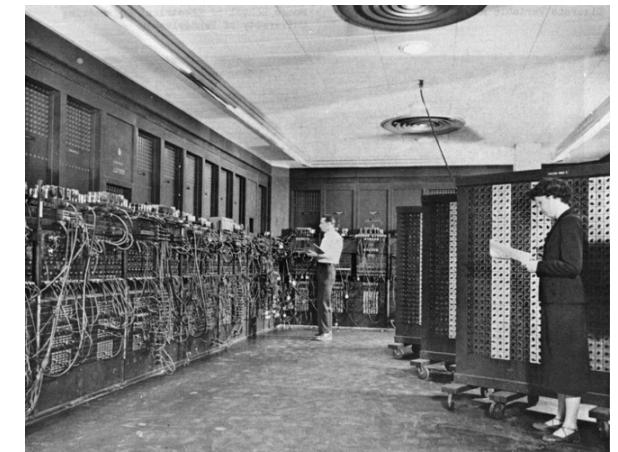
AUTREFOIS : ENIAC

- Premier ordinateur **entièlement électronique** :
 - 18 000 tubes à vide
 - 1 500 relais
 - 20 registres de 10 chiffres décimaux
 - programmé à l'aide de 6 000 commutateurs
- **La programmation** se faisait directement **en langage machine**



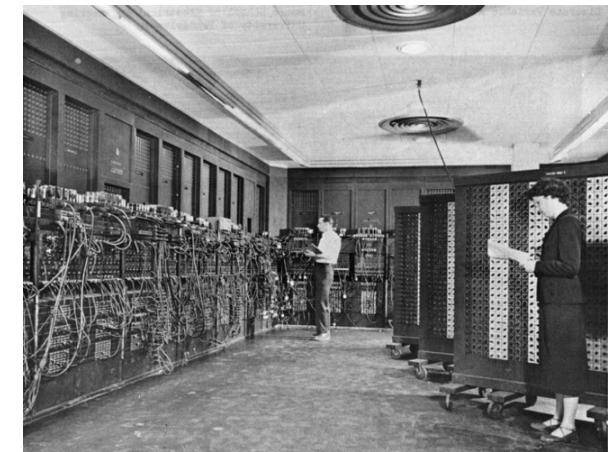
AUTREFOIS : ENIAC

- Premier ordinateur **entièlement électronique** :
 - 18 000 tubes à vide
 - 1 500 relais
 - 20 registres de 10 chiffres décimaux
 - programmé à l'aide de 6 000 commutateurs
- **La programmation** se faisait directement **en langage machine**
- **Un seul programme** à la fois pouvait s'exécuté.



AUTREFOIS : ENIAC

- Premier ordinateur **entièlement électronique** :
 - 18 000 tubes à vide
 - 1 500 relais
 - 20 registres de 10 chiffres décimaux
 - programmé à l'aide de 6 000 commutateurs
- **La programmation** se faisait directement **en langage machine**
- **Un seul programme** à la fois pouvait s'exécuté.
- L'absence d'un OS obligeait le programmeur à **charger manuellement le programme**



AUTREFOIS : IBM RAMAC 305



AUTREFOIS : IBM RAMAC 305

- Premier ordinateur à disque dur (l'**IBM 350**) commercialisé en septembre 1956 par **IBM**.



AUTREFOIS : IBM RAMAC 305



- Premier ordinateur à disque dur (l'**IBM 350**) commercialisé en **septembre 1956** par **IBM**.
- Composé des éléments suivants : unité de traitement, imprimante, console, alimentation, disque dur, mémoire 5Mo.

AUTREFOIS : IBM RAMAC 305



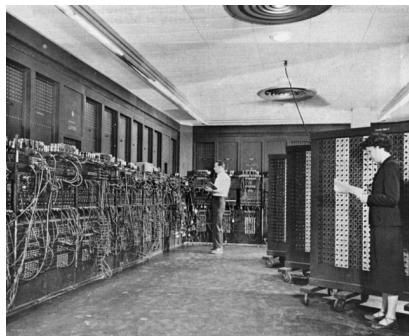
- Premier ordinateur à disque dur (l'**IBM 350**) commercialisé en **septembre 1956** par **IBM**.
- Composé des éléments suivants : unité de traitement, imprimante, console, alimentation, disque dur, mémoire 5Mo.
- **L'unité de traitement** est basée sur un tambour magnétique sur lequel est stocké le programme.

AUTREFOIS : IBM RAMAC 305

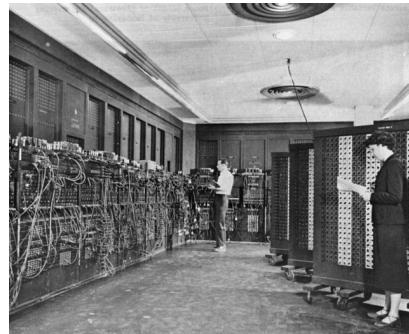


- Premier ordinateur à disque dur (l'**IBM 350**) commercialisé en **septembre 1956** par **IBM**.
- Composé des éléments suivants : unité de traitement, imprimante, console, alimentation, disque dur, mémoire 5Mo.
- L'**unité de traitement** est basée sur un tambour magnétique sur lequel est stocké le programme.
- Un opérateur programme à l'aide de **cartes perforées** et inscrit les données sur le tambour.

AUTOMATISER LES TÂCHES

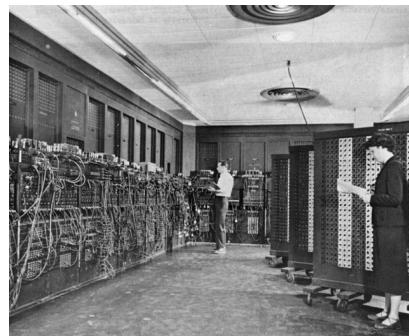


AUTOMATISER LES TÂCHES



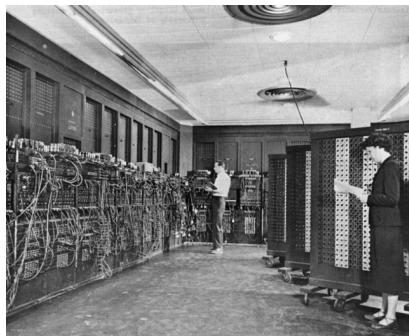
- Comment **automatiser les tâches** des opérateurs et des programmeurs ?

AUTOMATISER LES TÂCHES



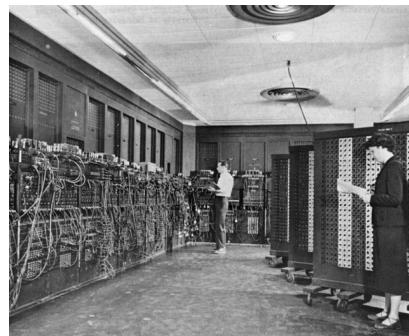
- Comment **automatiser les tâches** des opérateurs et des programmeurs ?
- Écrire un **programme informatique** qui:

AUTOMATISER LES TÂCHES



- Comment **automatiser les tâches** des opérateurs et des programmeurs ?
- Écrire un **programme informatique** qui:
👉 décide qui fait quoi et à quel moment

AUTOMATISER LES TÂCHES



- Comment **automatiser les tâches** des opérateurs et des programmeurs ?
- Écrire un **programme informatique** qui:
 - 👉 décide qui fait quoi et à quel moment
 - 👉 fait le lien entre les applications et le matériel

DÉFINITION

“Un système d'exploitation est un ensemble de programmes réalisant l'interface entre le matériel et les utilisateurs.”

Applications



Système d'exploitation



Matériel



DÉFINITION

“Un système d'exploitation est un ensemble de programmes réalisant l'interface entre le matériel et les utilisateurs.”

- gère la partie matérielle

Applications



Système d'exploitation



Matériel



DÉFINITION

“Un système d'exploitation est un ensemble de programmes réalisant l'interface entre le matériel et les utilisateurs.”

- gère la partie matérielle
- sert de socle pour les applications

Applications



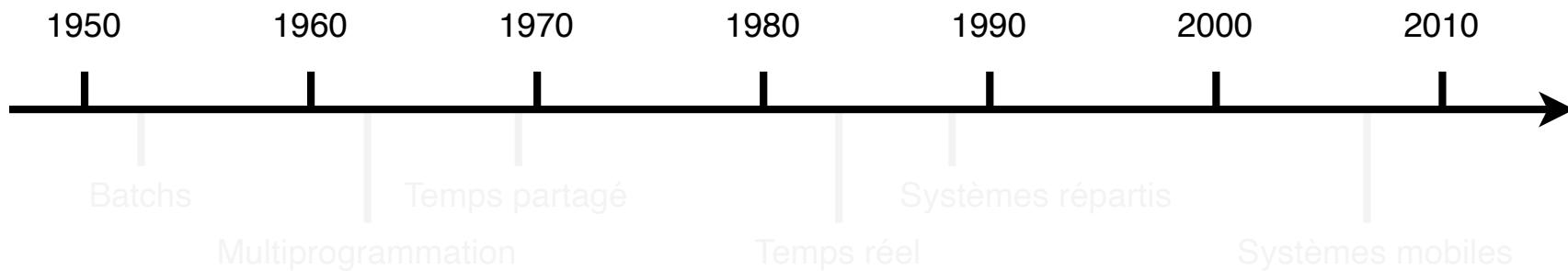
Système d'exploitation



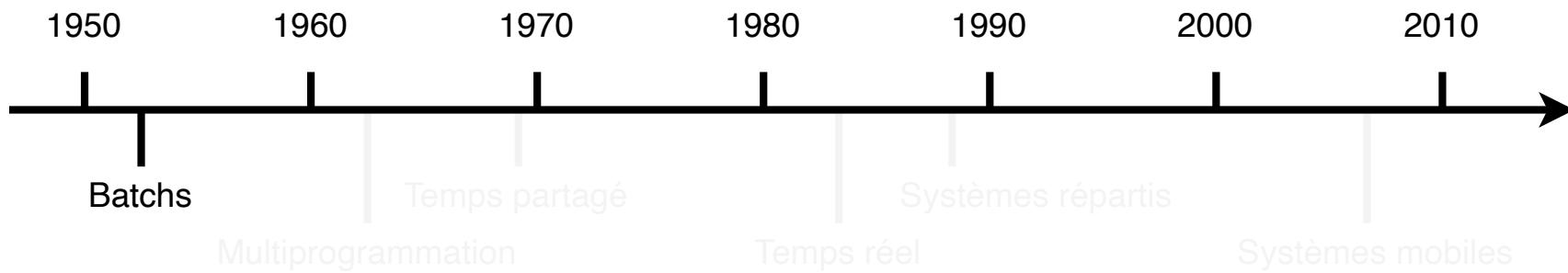
Matériel



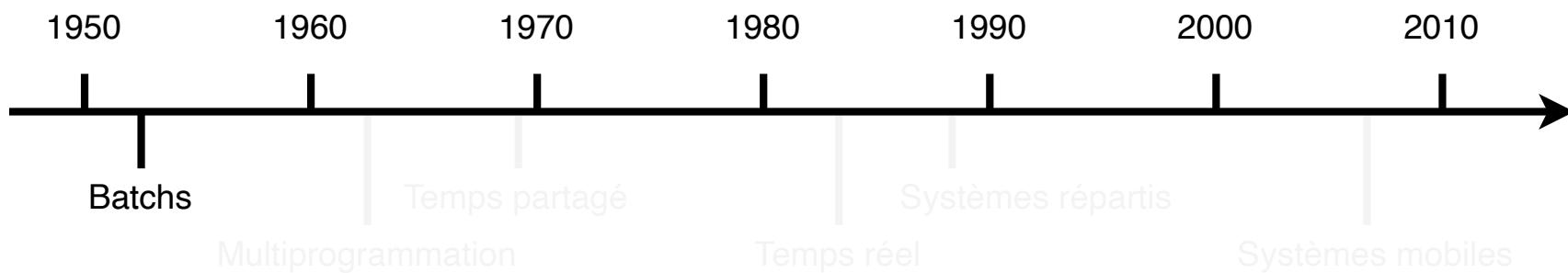
HISTORIQUE/TYPES DES OS



HISTORIQUE/TYPES DES OS

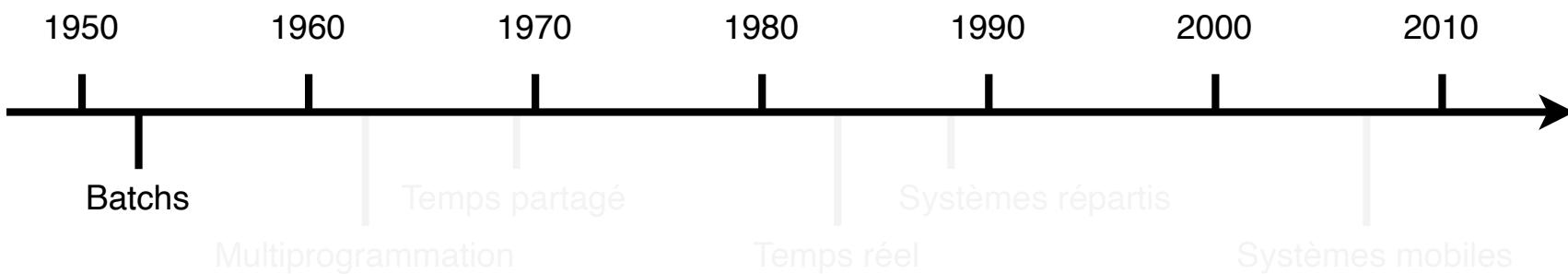


HISTORIQUE/TYPES DES OS



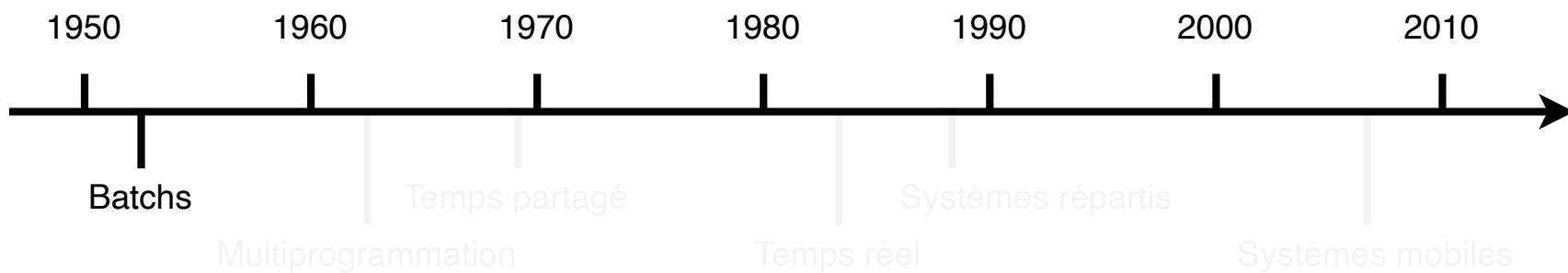
- **Les systèmes batch** sont basés sur deux programmes :

HISTORIQUE/TYPES DES OS



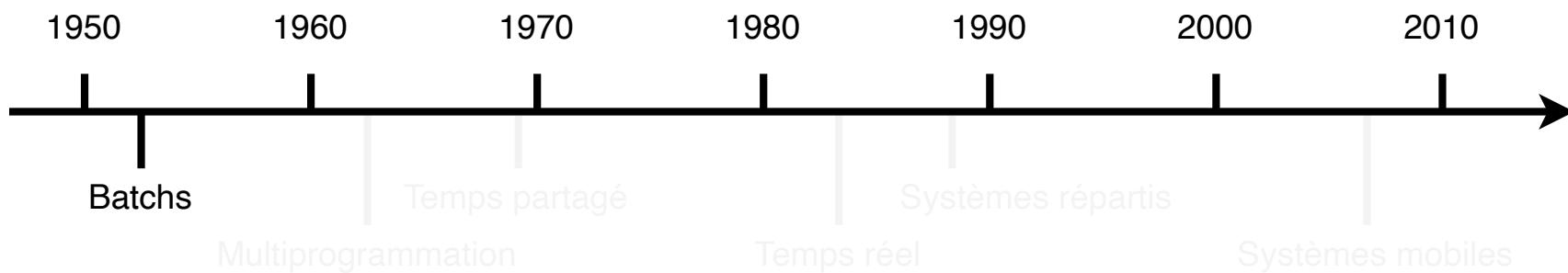
- **Les systèmes batch** sont basés sur deux programmes :
 1. **le chargeur** : charger les programmes dans la mémoire centrale depuis les cartes perforées ou le dérouleur de bandes.

HISTORIQUE/TYPES DES OS



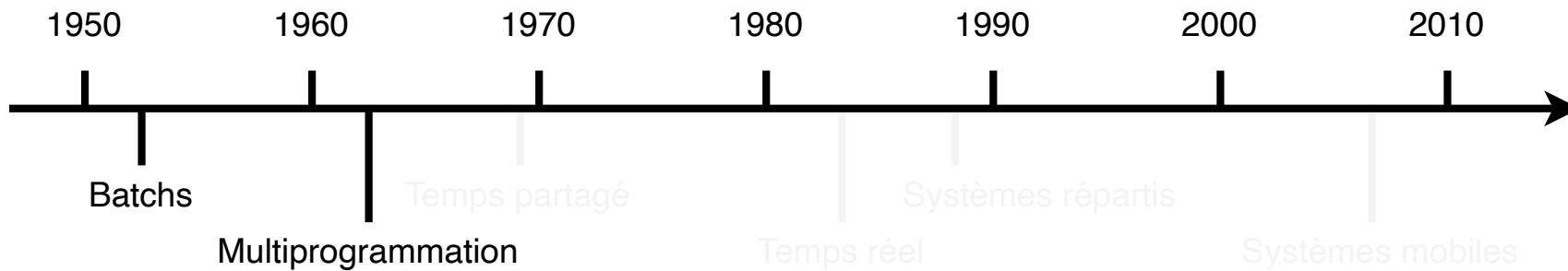
- **Les systèmes batch** sont basés sur deux programmes :
 1. **le chargeur** : charger les programmes dans la mémoire centrale depuis les cartes perforées ou le dérouleur de bandes.
 2. **le moniteur d'enchaînement de traitements** : permettre l'enchaînement des travaux soumis à la place de l'opérateur.

HISTORIQUE/TYPES DES OS

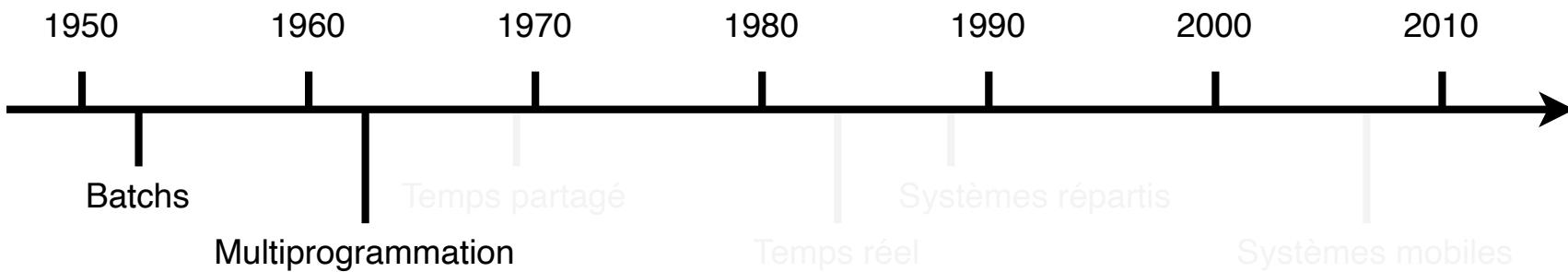


- **Les systèmes batch** sont basés sur deux programmes :
 1. **le chargeur** : charger les programmes dans la mémoire centrale depuis les cartes perforées ou le dérouleur de bandes.
 2. **le moniteur d'enchaînement de traitements** : permettre l'enchaînement des travaux soumis à la place de l'opérateur.
- **Les systèmes batch** automatisent les tâches de préparation des travaux et exploitent efficacement le processeur.

HISTORIQUE/TYPES DES OS

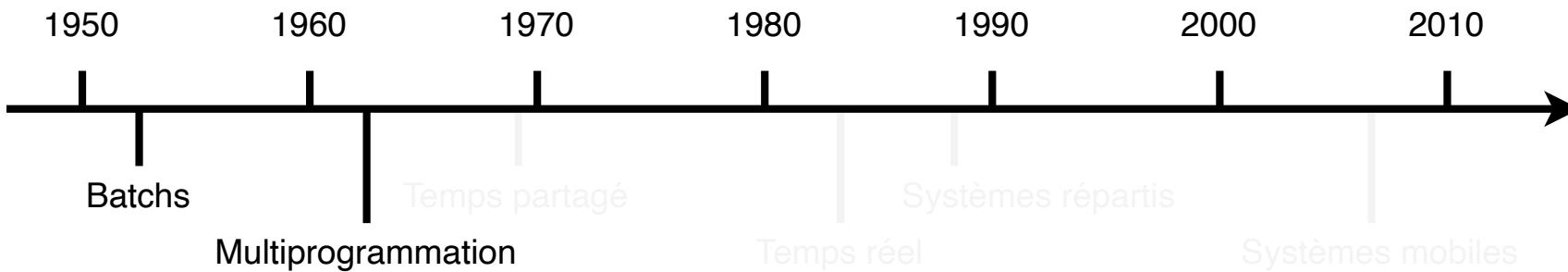


HISTORIQUE/TYPES DES OS



Utiliser plusieurs composants en parallèle, ce qui nécessite:

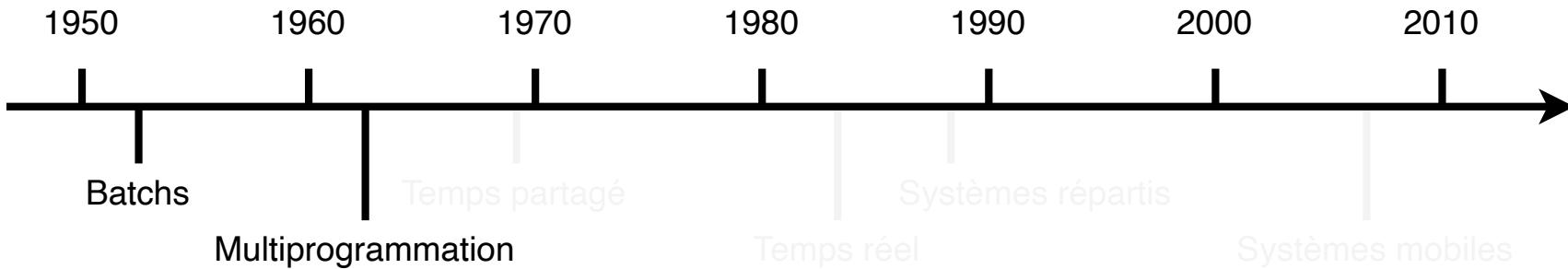
HISTORIQUE/TYPES DES OS



Utiliser plusieurs composants en parallèle, ce qui nécessite:

- **Gestion de la priorité** (*quel processus peut accéder à la ressource*)
👉 ordonnancement

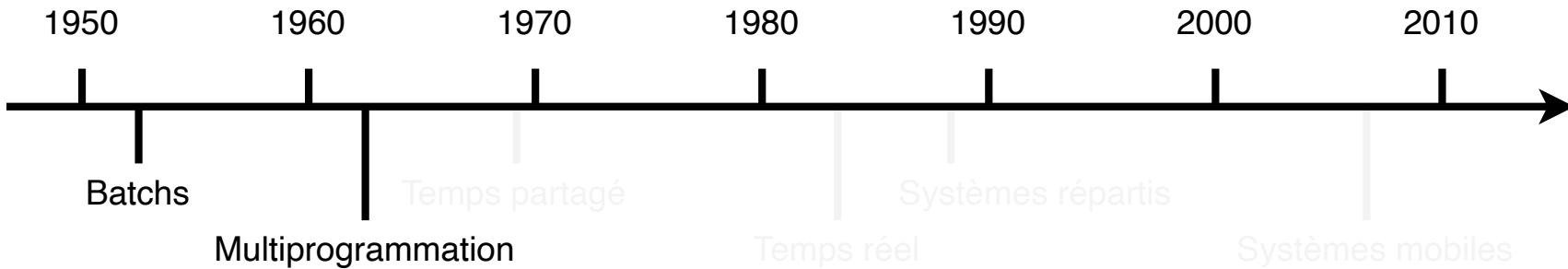
HISTORIQUE/TYPES DES OS



Utiliser plusieurs composants en parallèle, ce qui nécessite:

- **Gestion de la priorité** (*quel processus peut accéder à la ressource*)
👉 ordonnancement
- **Mémoire partagée** (*gérer des informations de plusieurs processus*)
👉 adressage et mémoire

HISTORIQUE/TYPES DES OS

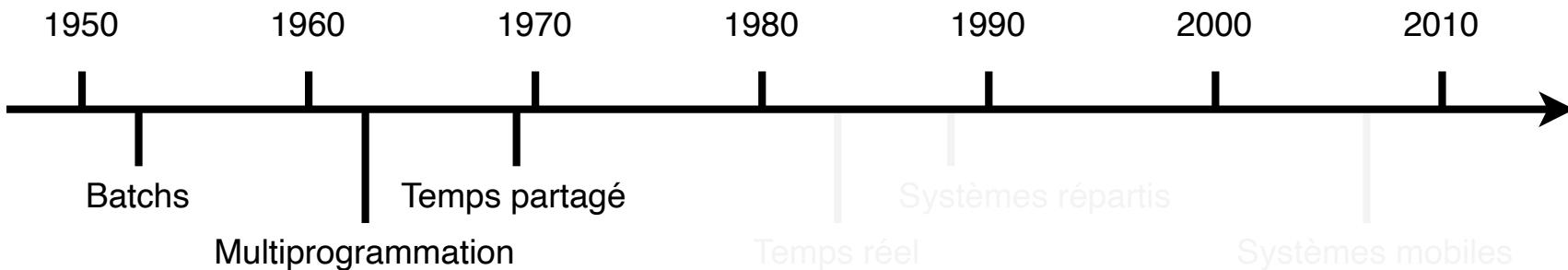


Utiliser plusieurs composants en parallèle, ce qui nécessite:

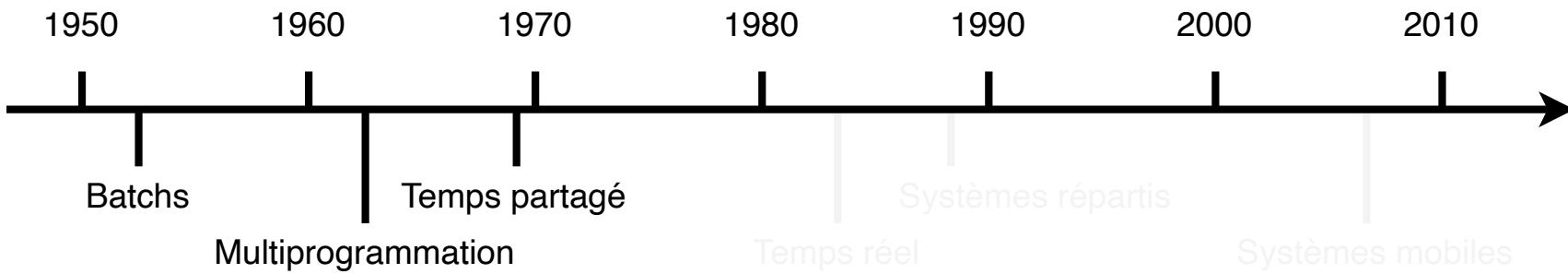
- **Gestion de la priorité** (*quel processus peut accéder à la ressource*)
👉 ordonnancement
- **Mémoire partagée** (*gérer des informations de plusieurs processus*)
👉 adressage et mémoire

Exemple : MULTICS

HISTORIQUE/TYPES DES OS

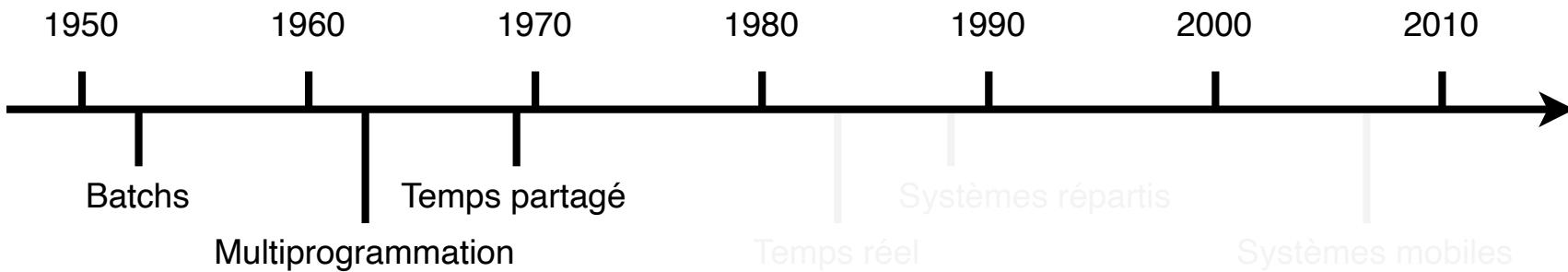


HISTORIQUE/TYPES DES OS



Plusieurs processus actifs alternant sur le processeur

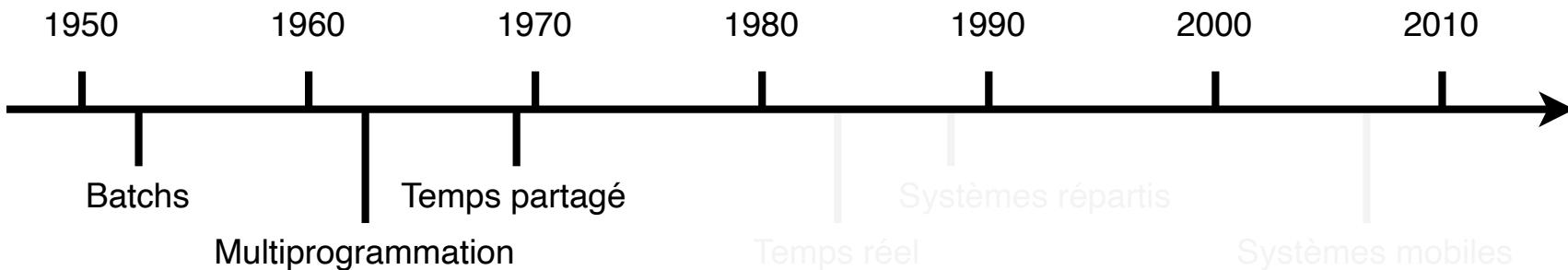
HISTORIQUE/TYPES DES OS



Plusieurs processus actifs alternant sur le processeur

- Gestion des interruptions

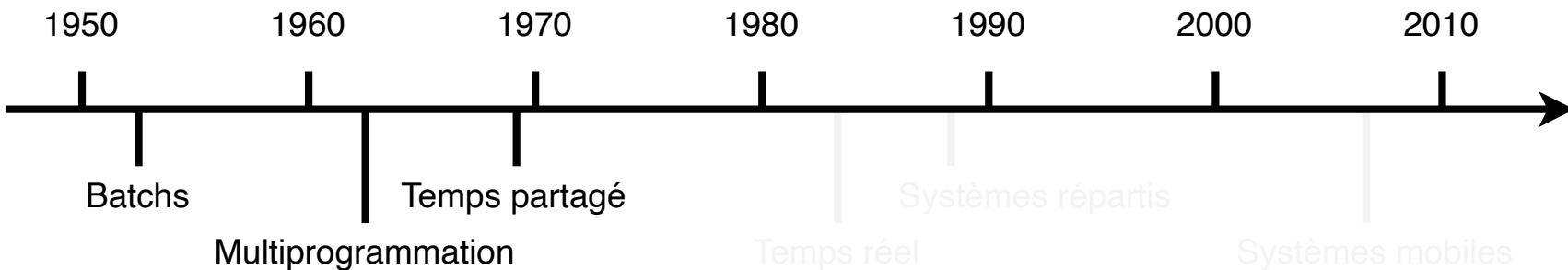
HISTORIQUE/TYPES DES OS



Plusieurs processus actifs alternant sur le processeur

- Gestion des interruptions
- Cycle de vie du processus

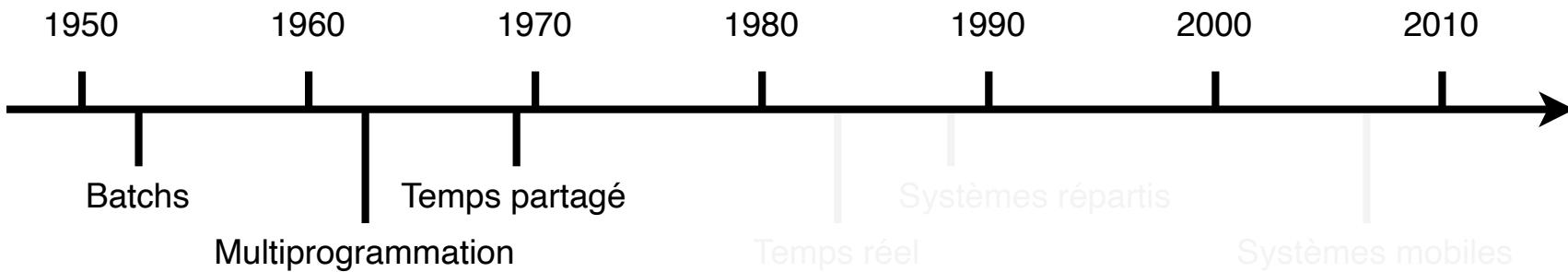
HISTORIQUE/TYPES DES OS



Plusieurs processus actifs alternant sur le processeur

- Gestion des interruptions
- Cycle de vie du processus
- Synchronisation de processus et programmation concurrente

HISTORIQUE/TYPES DES OS

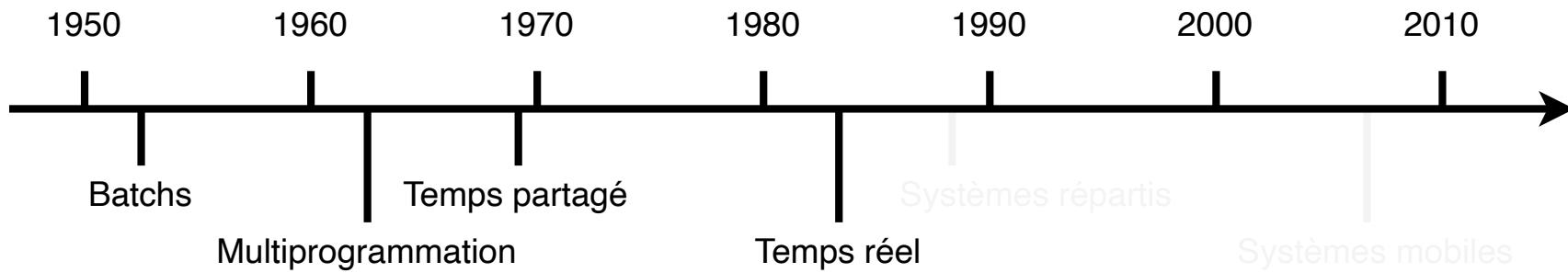


Plusieurs processus actifs alternant sur le processeur

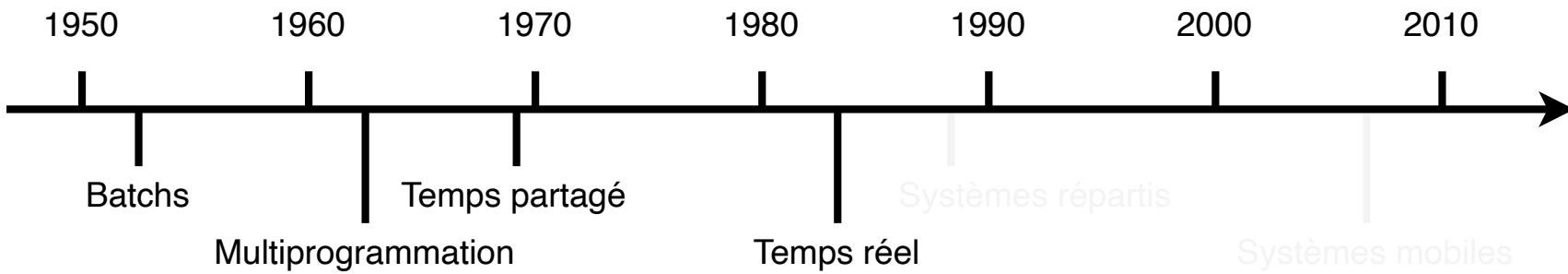
- Gestion des interruptions
- Cycle de vie du processus
- Synchronisation de processus et programmation concurrente

Exemple : UNICS ou UNIX

HISTORIQUE/TYPES DES OS

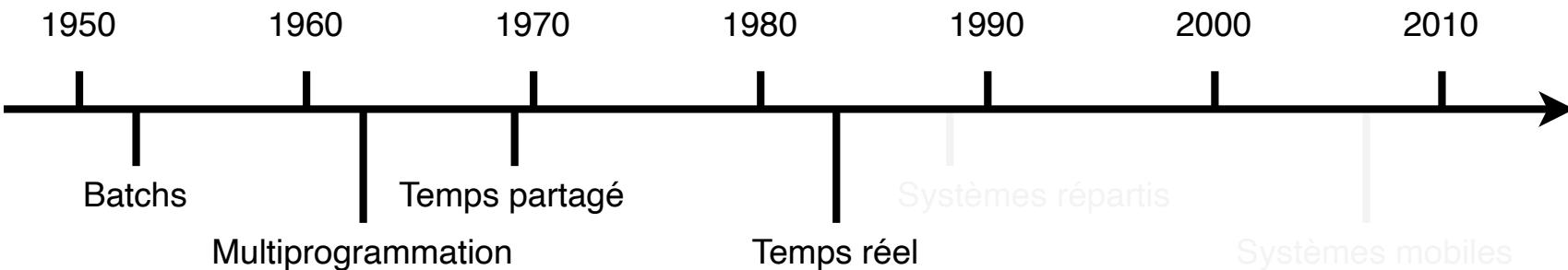


HISTORIQUE/TYPES DES OS



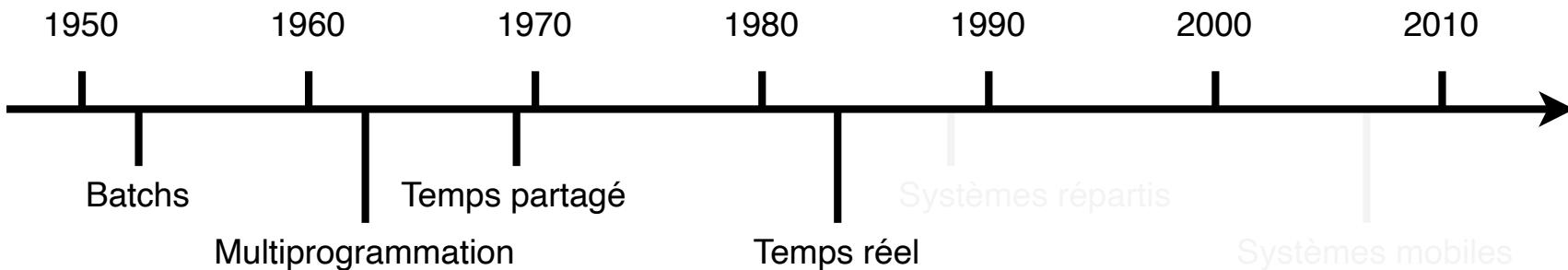
- **Gestion des délais:** contrainte de **temps de réponse**
👉 les processus doivent répondre vite

HISTORIQUE/TYPES DES OS



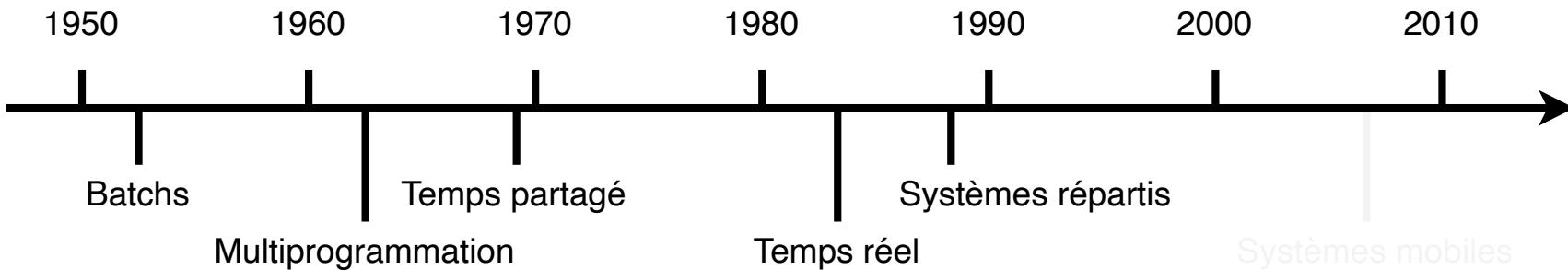
- **Gestion des délais:** contrainte de **temps de réponse**
👉 les processus doivent répondre vite
- Apparition des **micro-ordinateurs**
 - CP/M → IBM PC (MSDOS)

HISTORIQUE/TYPES DES OS

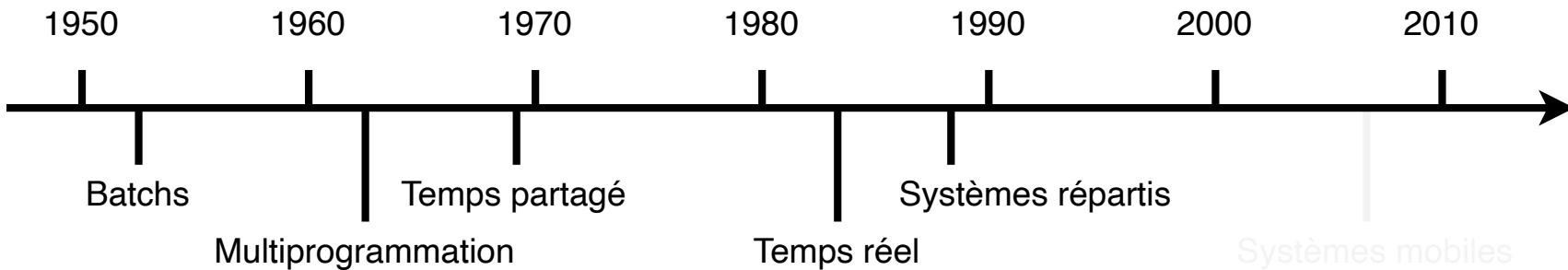


- **Gestion des délais:** contrainte de **temps de réponse**
👉 les processus doivent répondre vite
- Apparition des **micro-ordinateurs**
 - CP/M → IBM PC (MSDOS)
- Apparition des **interfaces graphiques**
 - Xerox → Apple Macintosh 1984, Windows 95, Linux 1991

HISTORIQUE/TYPES DES OS

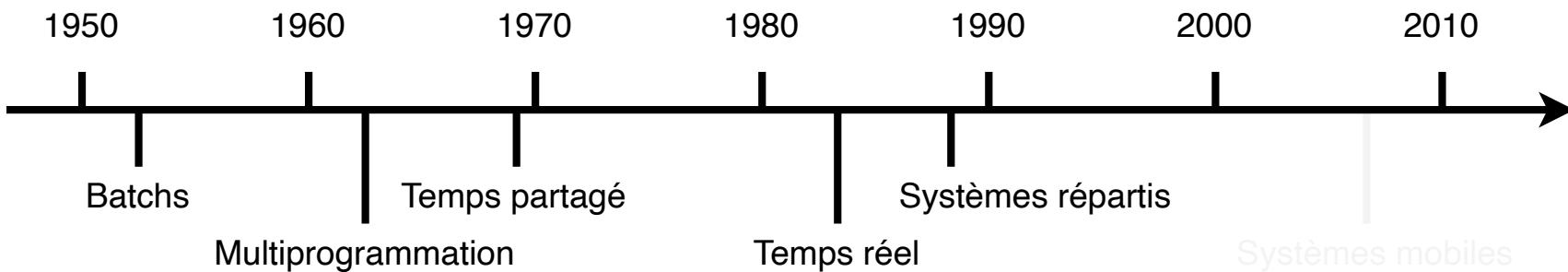


HISTORIQUE/TYPES DES OS



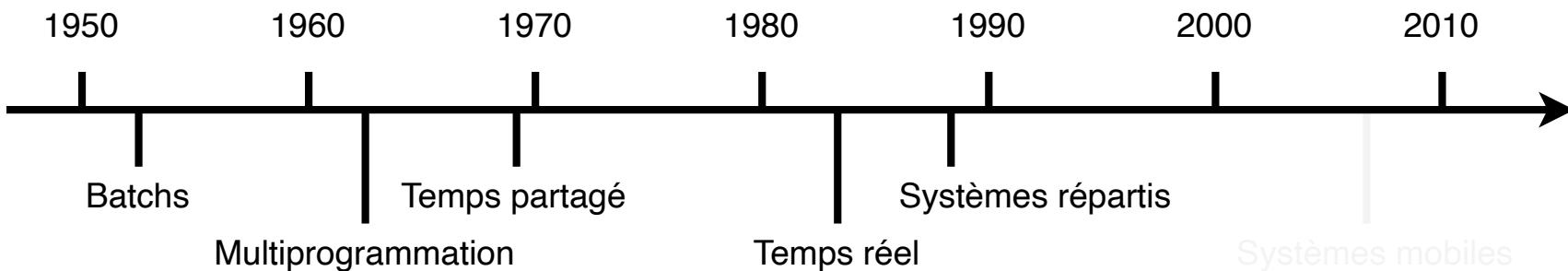
- **Les ordinateurs communiquent** pour échanger des données !

HISTORIQUE/TYPES DES OS



- **Les ordinateurs communiquent** pour échanger des données !
 - Arpanet (1967) conçu par la DARPA

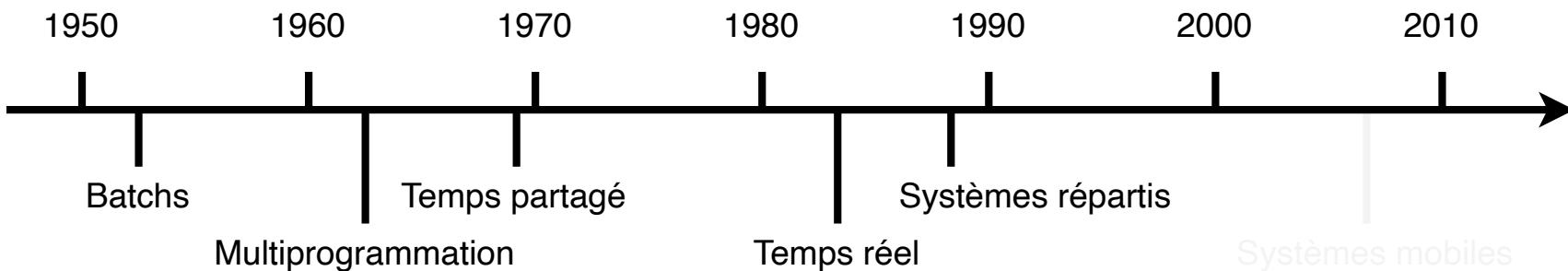
HISTORIQUE/TYPES DES OS



- **Les ordinateurs communiquent** pour échanger des données !

- Arpanet (1967) conçu par la DARPA
- E-mail (1972) avec Ray Tomlinson

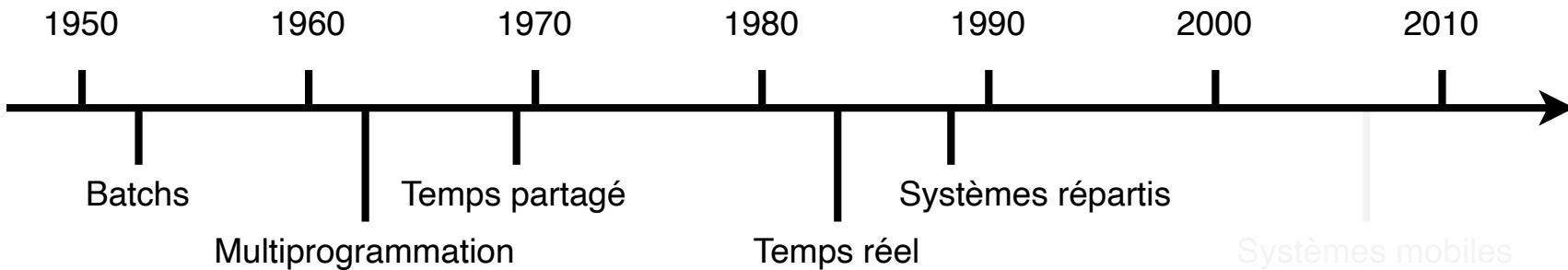
HISTORIQUE/TYPES DES OS



- **Les ordinateurs communiquent** pour échanger des données !

- Arpanet (1967) conçu par la DARPA
- E-mail (1972) avec Ray Tomlinson
- TCP/IP(1972)

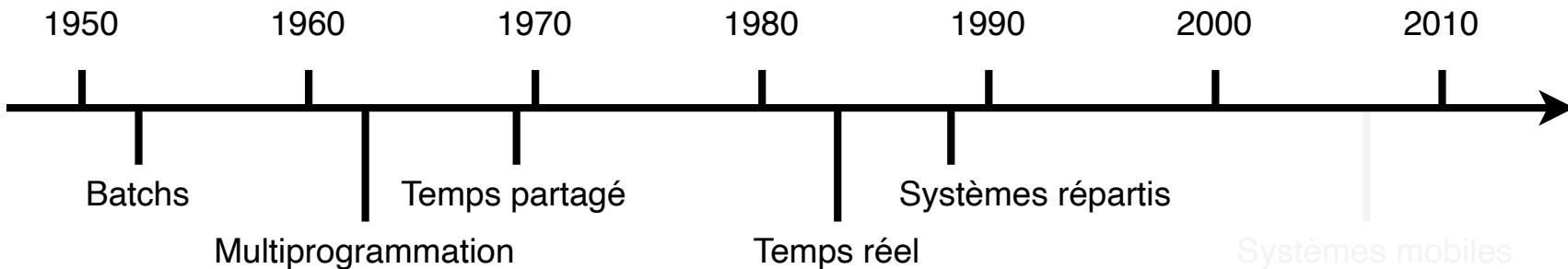
HISTORIQUE/TYPES DES OS



- **Les ordinateurs communiquent** pour échanger des données !

- Arpanet (1967) conçu par la DARPA
- E-mail (1972) avec Ray Tomlinson
- TCP/IP(1972)
- Clients-Serveur années 80 → NFS - Network File System (Sun, 1984)

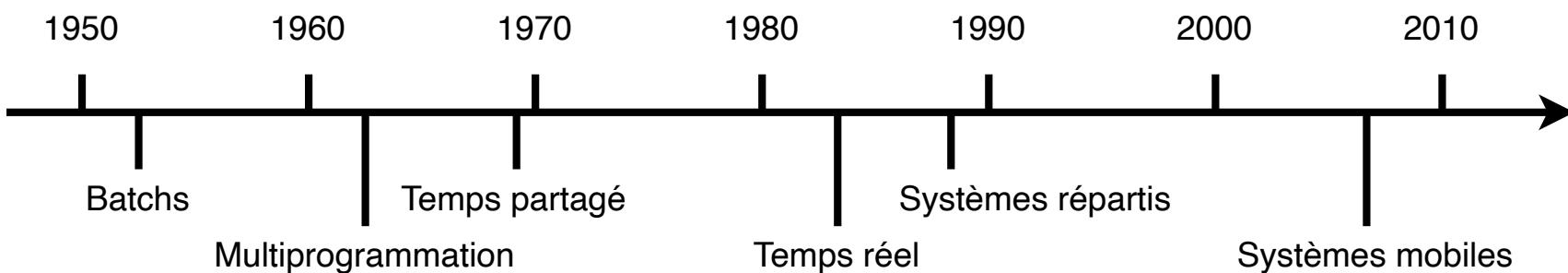
HISTORIQUE/TYPES DES OS



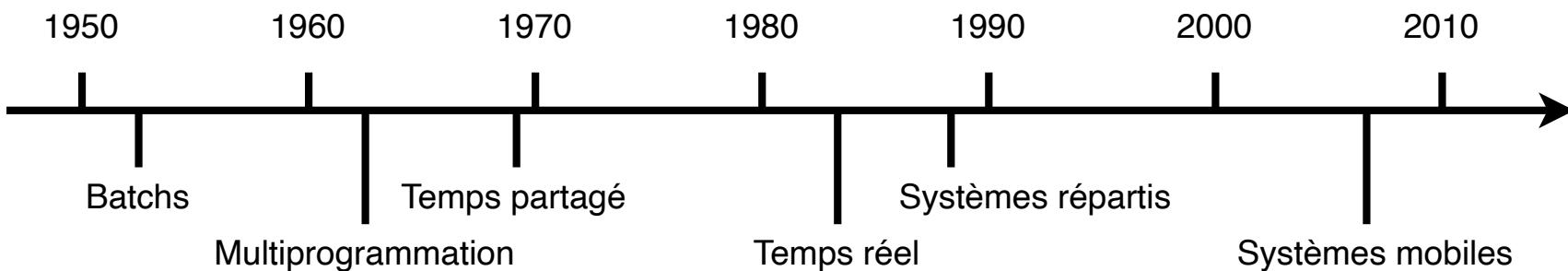
- **Les ordinateurs communiquent** pour échanger des données !

- Arpanet (1967) conçu par la DARPA
- E-mail (1972) avec Ray Tomlinson
- TCP/IP(1972)
- Clients-Serveur années 80 → NFS - Network File System (Sun, 1984)
- Arpanet ouvert fin 80 → Web début 90 (CERN , Tim Berners-Lee)

HISTORIQUE/TYPES DES OS

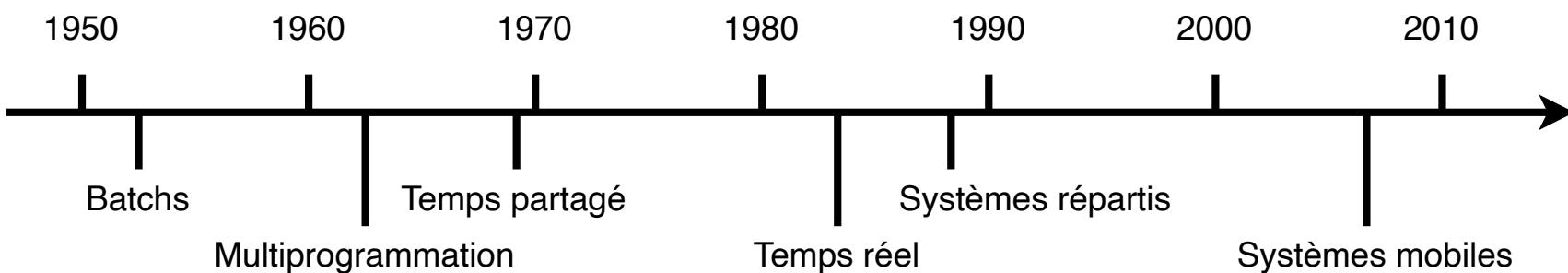


HISTORIQUE/TYPES DES OS



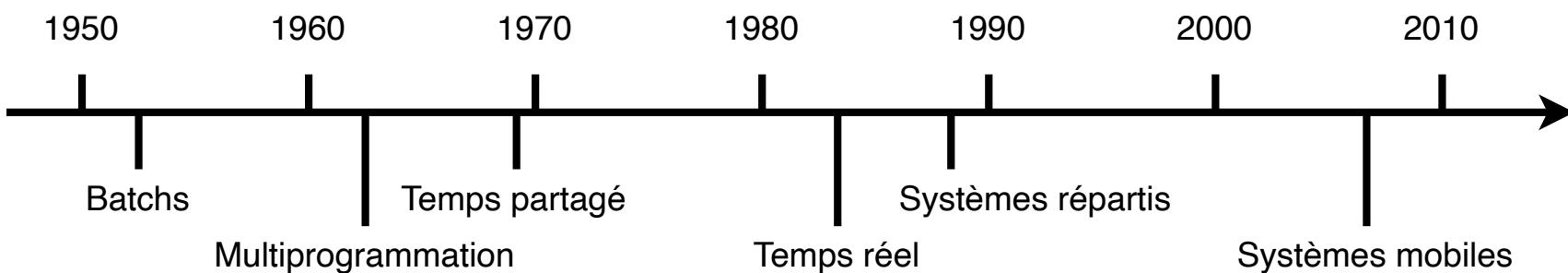
- **Les ordinateurs de poche** existent depuis les années 80

HISTORIQUE/TYPES DES OS



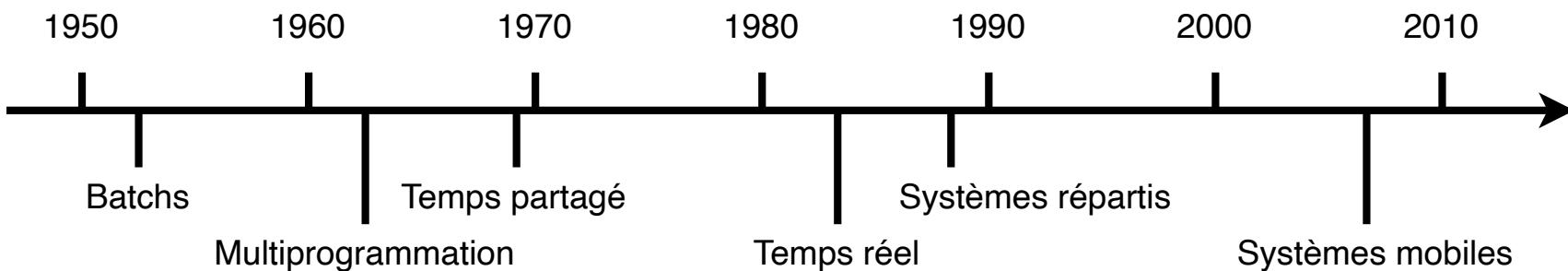
- **Les ordinateurs de poche** existent depuis les années 80
 - 1986 : sortie des PDA → PalmOS

HISTORIQUE/TYPES DES OS



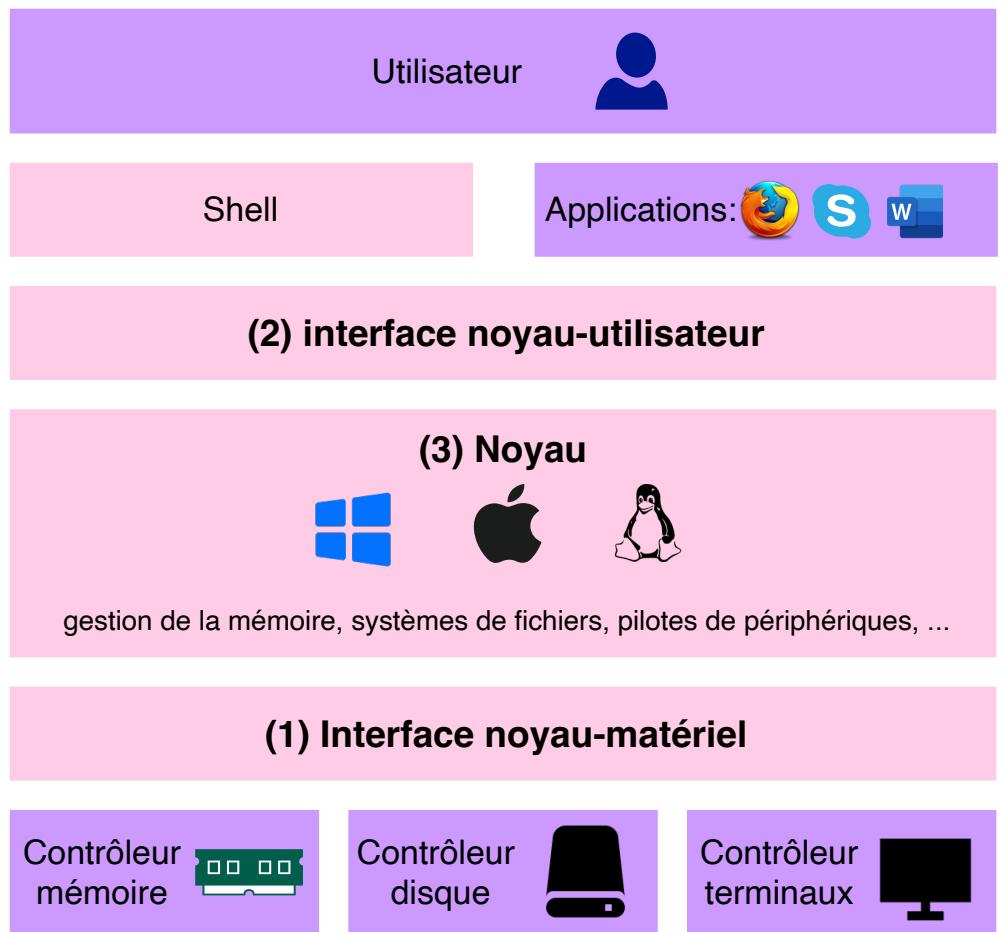
- **Les ordinateurs de poche** existent depuis les années 80
 - 1986 : sortie des PDA → PalmOS
 - 2007 : sortie des smartphones → android OS

HISTORIQUE/TYPES DES OS



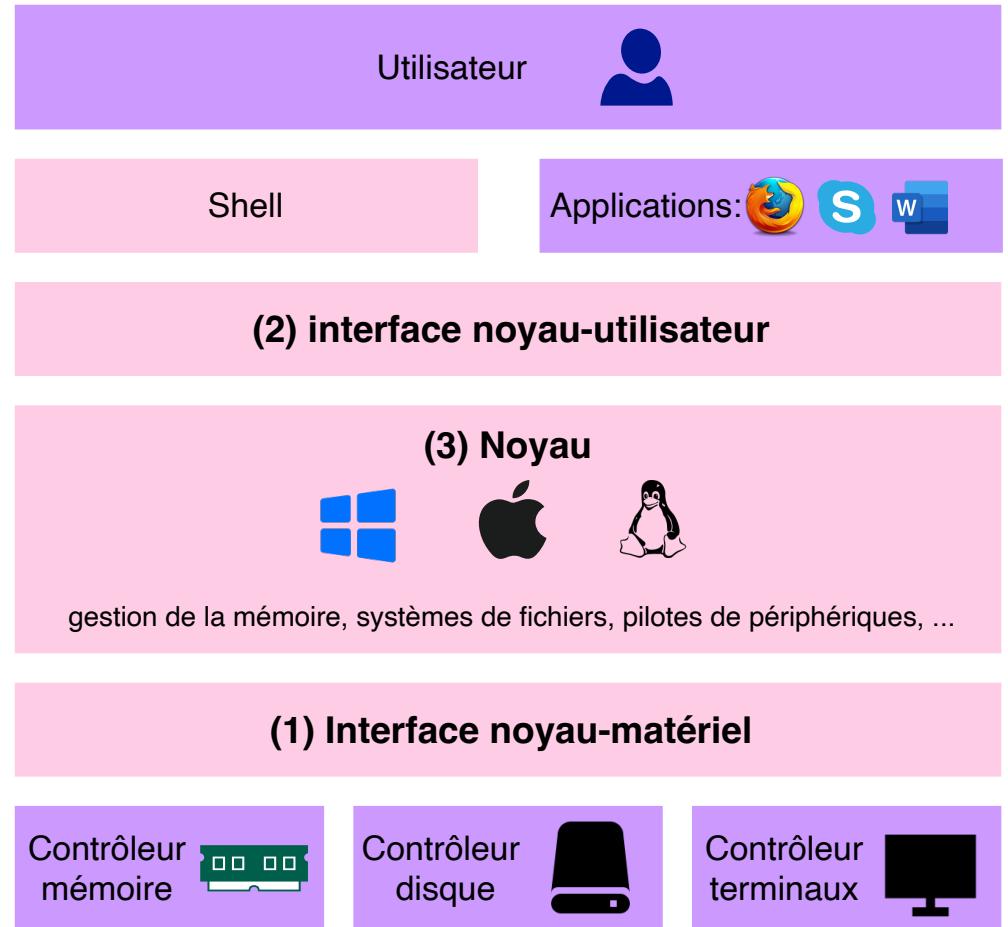
- **Les ordinateurs de poche** existent depuis les années 80
 - 1986 : sortie des PDA → PalmOS
 - 2007 : sortie des smartphones → android OS
 - 2007 : sortie de l'iPhone → iOS

RÔLES DU SYSTÈME D'EXPLOITATION



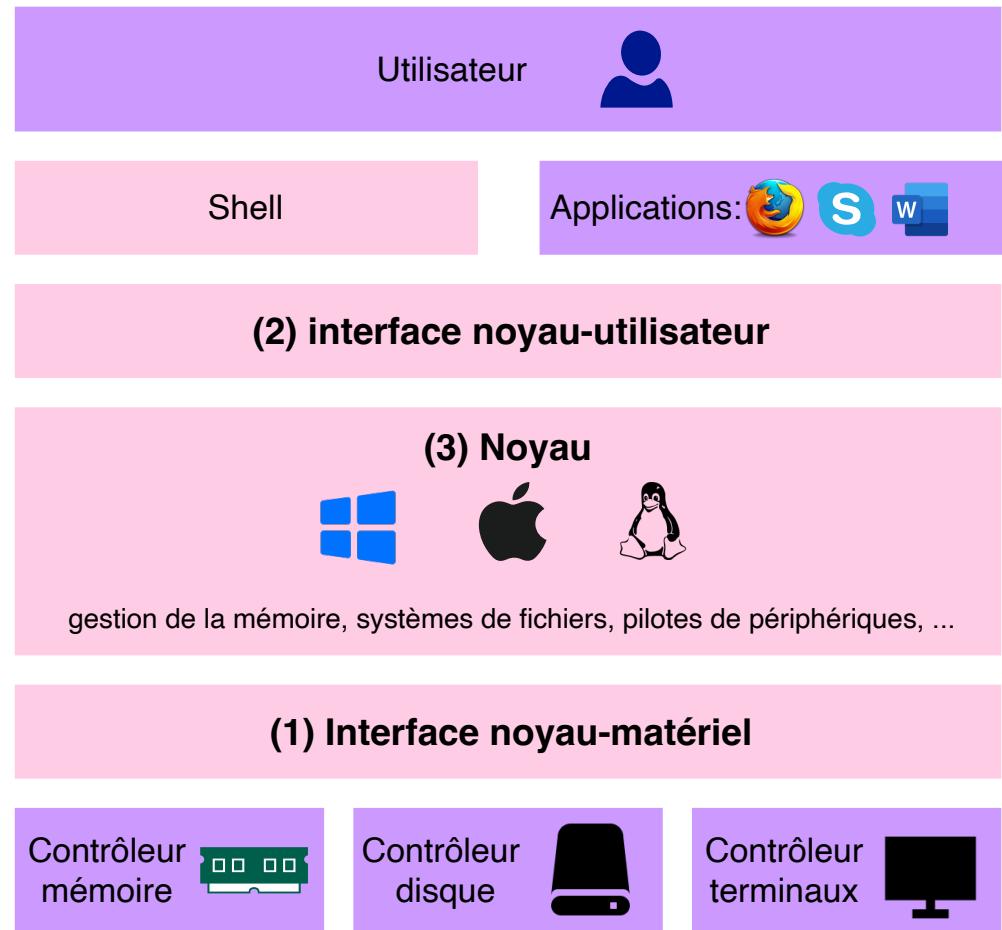
RÔLES DU SYSTÈME D'EXPLOITATION

1. L'interface noyau-matériel prend en charge la gestion et le partage des ressources de la machine.



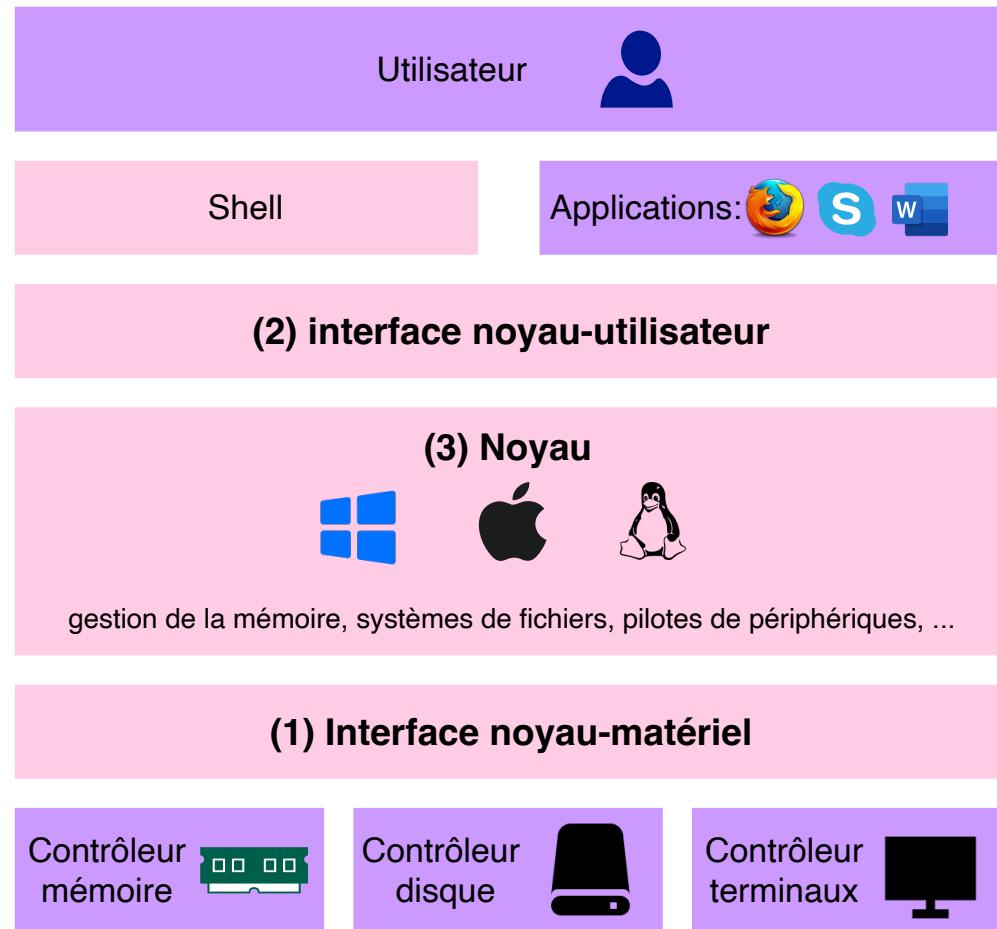
RÔLES DU SYSTÈME D'EXPLOITATION

1. L'**interface noyau-matériel** prend en charge la gestion et le partage des ressources de la machine.
2. L'**interface noyau-utilisateur** construit une machine virtuelle plus facile d'emploi et plus conviviale.

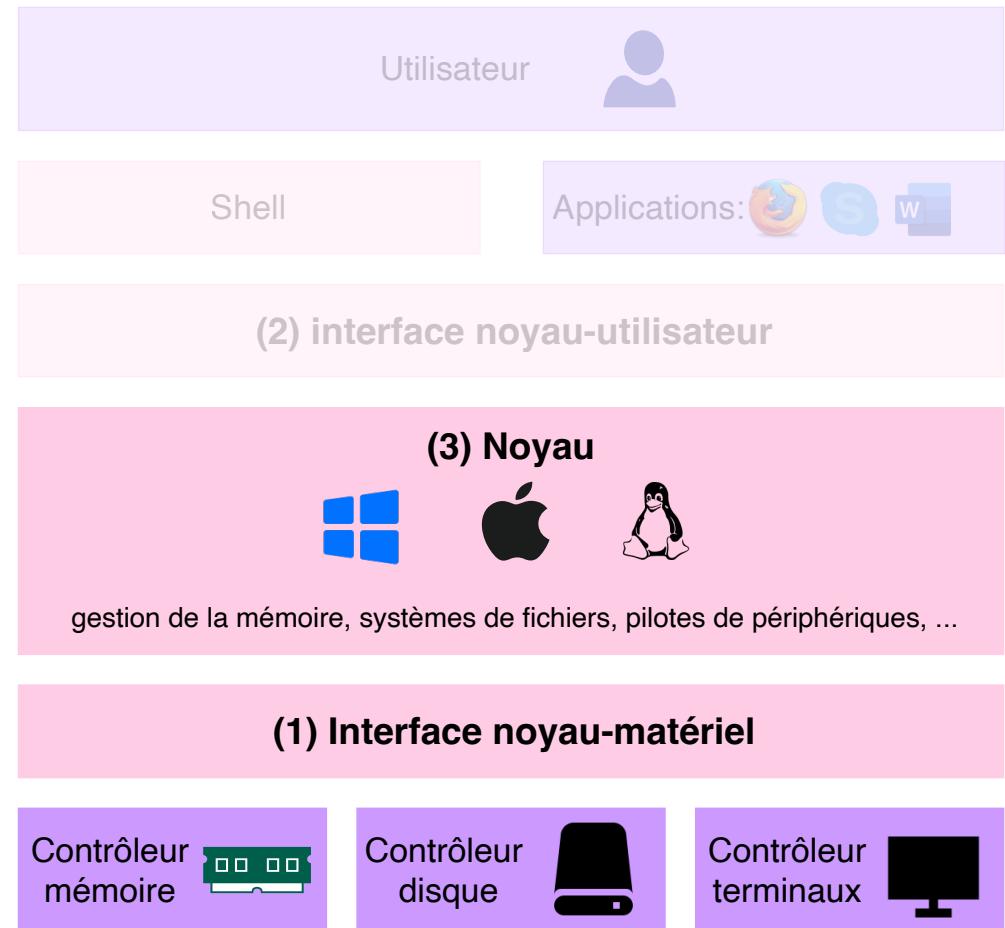


RÔLES DU SYSTÈME D'EXPLOITATION

1. **L'interface noyau-matériel** prend en charge la gestion et le partage des ressources de la machine.
2. **L'interface noyau-utilisateur** construit une machine virtuelle plus facile d'emploi et plus conviviale.
3. **Le noyau** assure plusieurs grandes fonctionnalités.



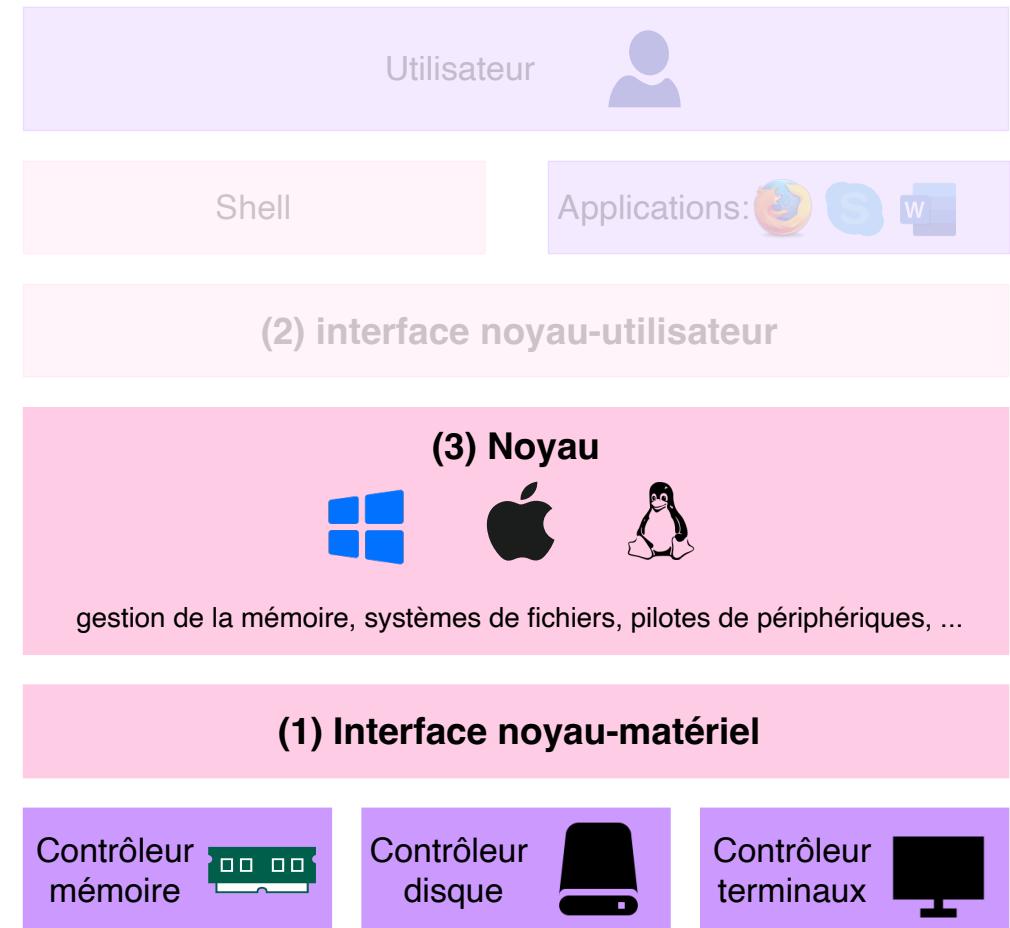
INTERFACE NOYAU-MATÉRIEL



INTERFACE NOYAU-MATÉRIEL

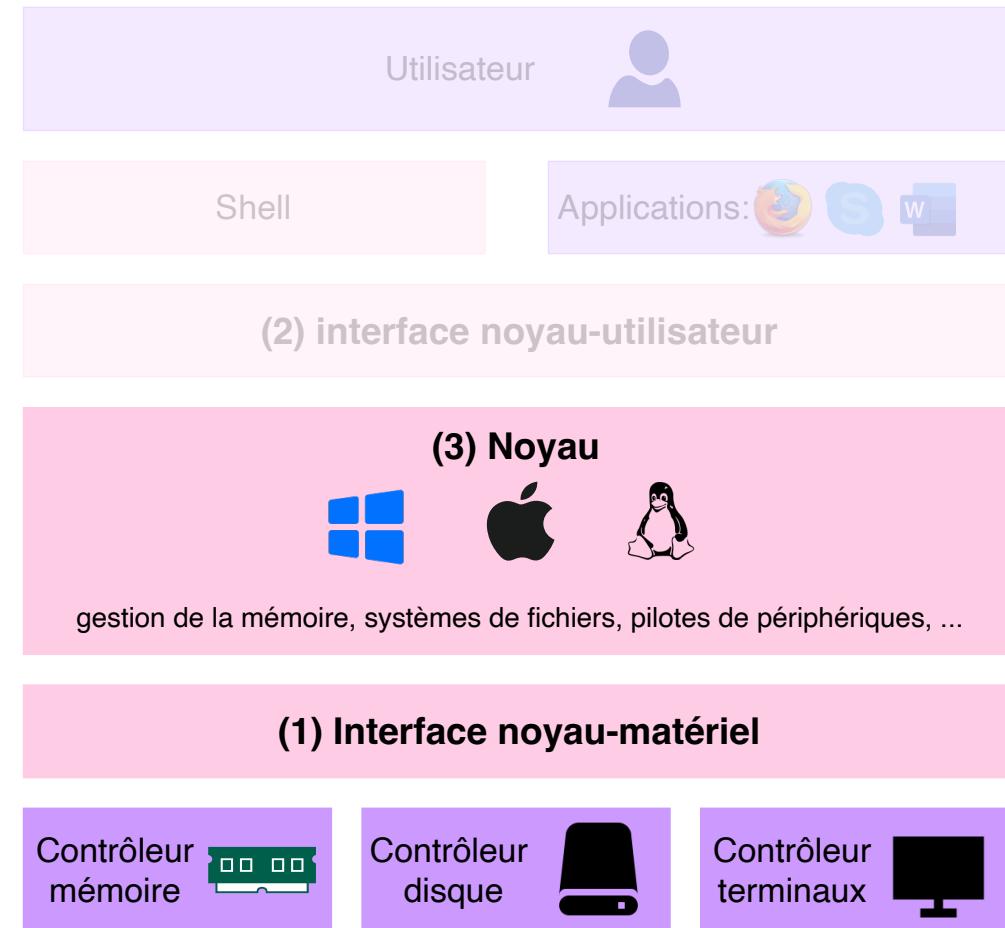
- Gérer l'accès et le partage des ressources matérielles (arbitrage).

- processeur
- mémoire centrale
- périphériques
- ...



INTERFACE NOYAU-MATÉRIEL

- Gérer l'accès et le partage des ressources matérielles (arbitrage).
 - processeur
 - mémoire centrale
 - périphériques
 - ...
- Cet arbitrage doit assurer:

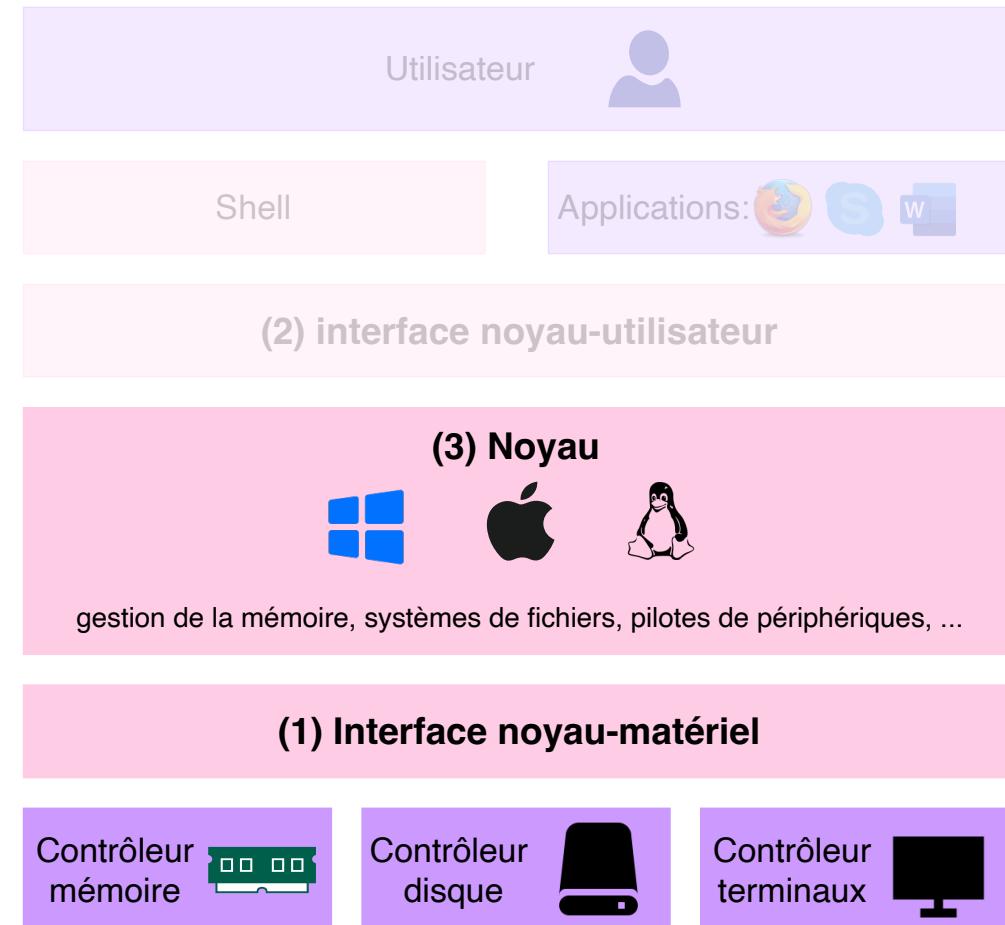


INTERFACE NOYAU-MATÉRIEL

- Gérer l'accès et le partage des ressources matérielles (arbitrage).

- processeur
- mémoire centrale
- périphériques
- ...

- Cet arbitrage doit assurer:
 - l'équité d'accès aux ressources

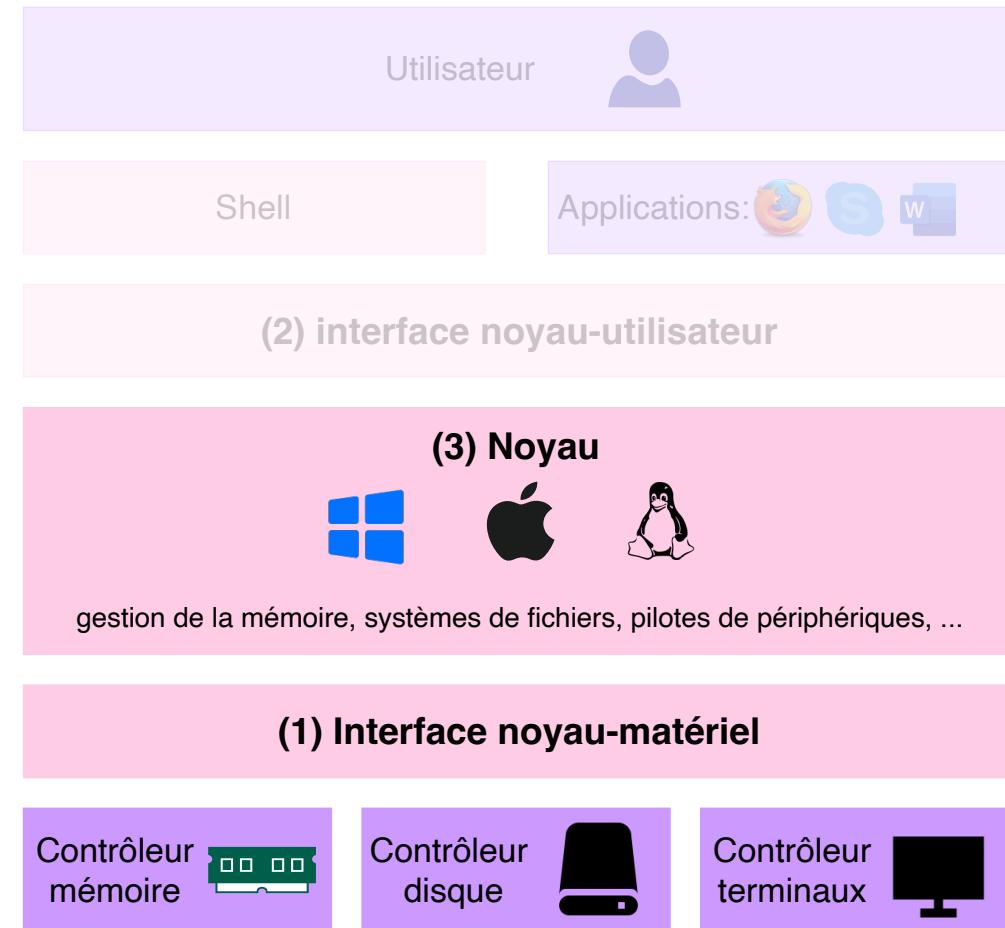


INTERFACE NOYAU-MATÉRIEL

- Gérer l'accès et le partage des ressources matérielles (arbitrage).

- processeur
- mémoire centrale
- périphériques
- ...

- Cet arbitrage doit assurer:
 - l'équité d'accès aux ressources
 - la protection de l'accès aux ressources



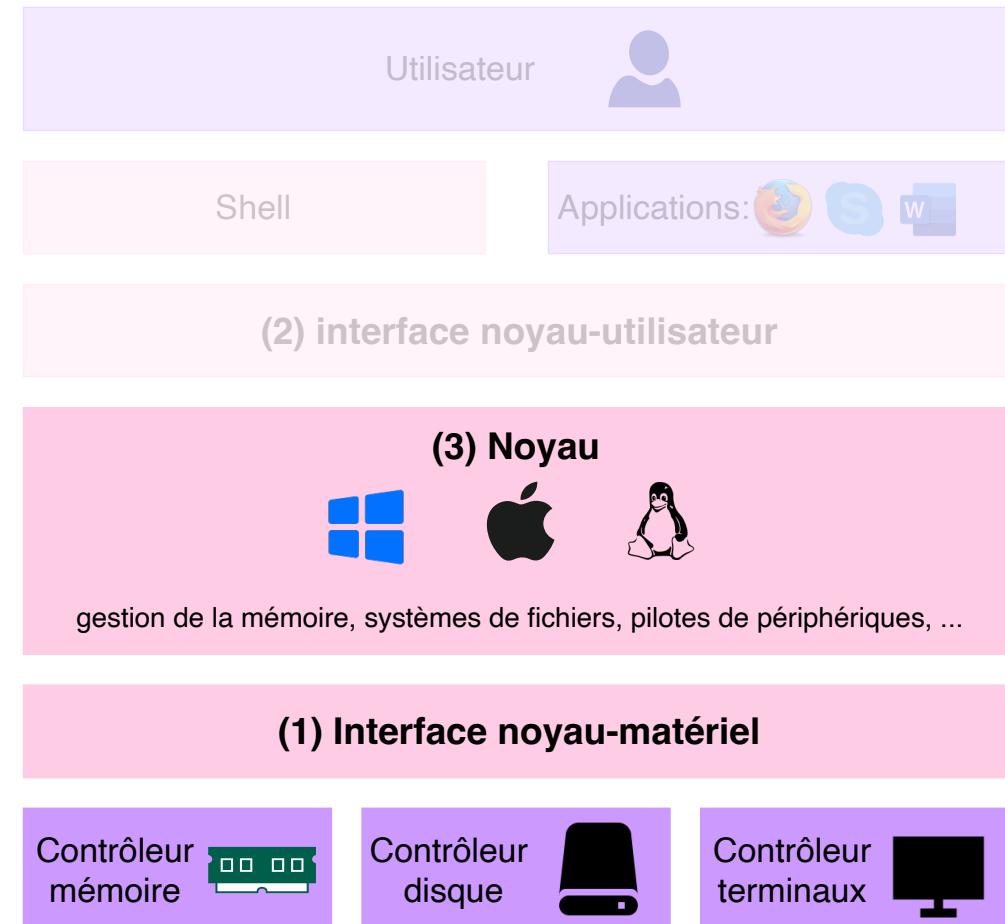
INTERFACE NOYAU-MATÉRIEL

- Gérer l'accès et le partage des ressources matérielles (arbitrage).

- processeur
- mémoire centrale
- périphériques
- ...

- Cet arbitrage doit assurer:

- l'équité d'accès aux ressources
- la protection de l'accès aux ressources
- la cohérence des états des ressources



INTERRUPTION - IRQ

INTERRUPTION - IRQ

- L'OS s'interface avec la couche matérielle, par le biais du mécanisme des interruptions (**Interrupt ReQuest ou IRQ**).

INTERRUPTION - IRQ

- L'OS s'interface avec la couche matérielle, par le biais du mécanisme des interruptions (**Interrupt ReQuest ou IRQ**).
👉 prendre connaissance des événements survenant sur le matérielle

INTERRUPTION - IRQ

- L'OS s'interface avec la couche matérielle, par le biais du mécanisme des interruptions (**Interrupt ReQuest ou IRQ**).
👉 prendre connaissance des événements survenant sur le matérielle
- L'**IRQ** est un signal (code) permettant à un dispositif externe d'interrompre le processeur pour lancer un traitement particulier.

INTERRUPTION - IRQ

- L'OS s'interface avec la couche matérielle, par le biais du mécanisme des **interruptions (Interrupt ReQuest ou IRQ)**.
👉 prendre connaissance des événements survenant sur le matérielle
- L'**IRQ** est un **signal (code)** permettant à un dispositif externe d'interrompre le processeur pour lancer un traitement particulier.
 - À chaque code correspond une **routine de traitement** de l'OS.

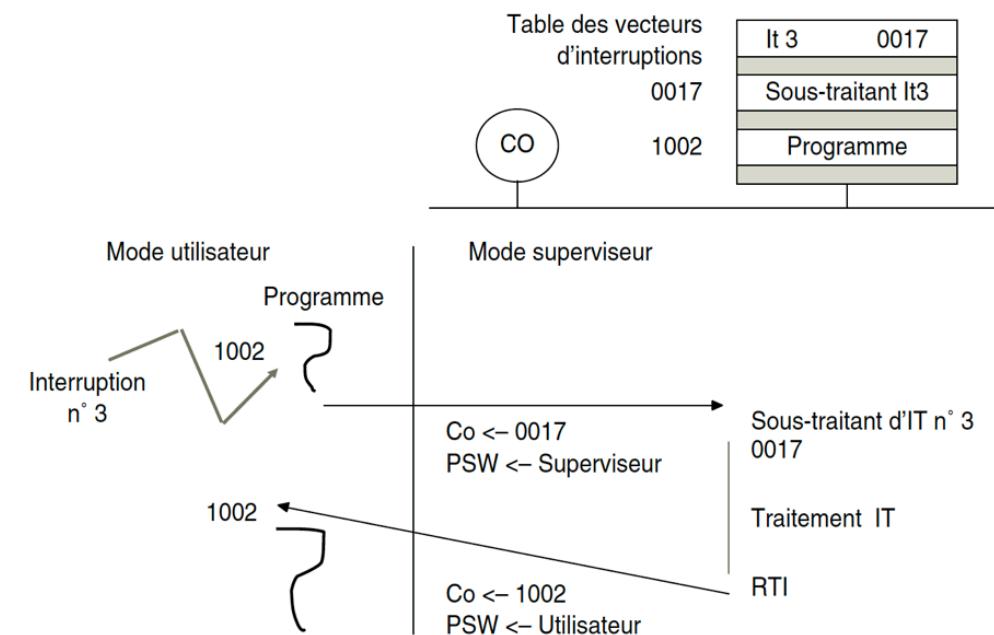
INTERRUPTION - IRQ

- L'OS s'interface avec la couche matérielle, par le biais du mécanisme des **interruptions (Interrupt ReQuest ou IRQ)**.
👉 prendre connaissance des événements survenant sur le matérielle
- L'**IRQ** est un **signal (code)** permettant à un dispositif externe d'interrompre le processeur pour lancer un traitement particulier.
 - À chaque code correspond une **routine de traitement** de l'OS.
 - Les adresses des routines sont dans une **table** placée en mémoire (**la table des vecteurs d'interruptions**).

INTERRUPTION - IRQ

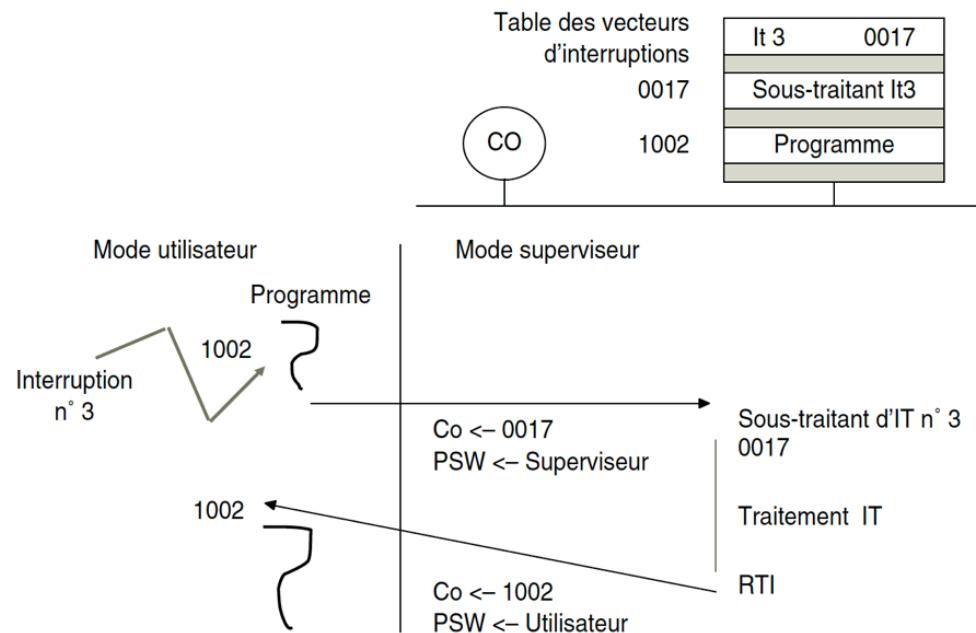
- L'OS s'interface avec la couche matérielle, par le biais du mécanisme des **interruptions (Interrupt ReQuest ou IRQ)**.
👉 prendre connaissance des événements survenant sur le matérielle
- L'**IRQ** est un **signal (code)** permettant à un dispositif externe d'interrompre le processeur pour lancer un traitement particulier.
 - À chaque code correspond une **routine de traitement** de l'OS.
 - Les adresses des routines sont dans une **table** placée en mémoire (**la table des vecteurs d'interruptions**).
 - **Les routines d'interruptions** sont chargées en mémoire au moment du chargement de l'OS et exécutées en **mode superviseur**.

PRISE EN COMPTE D'UNE IRQ



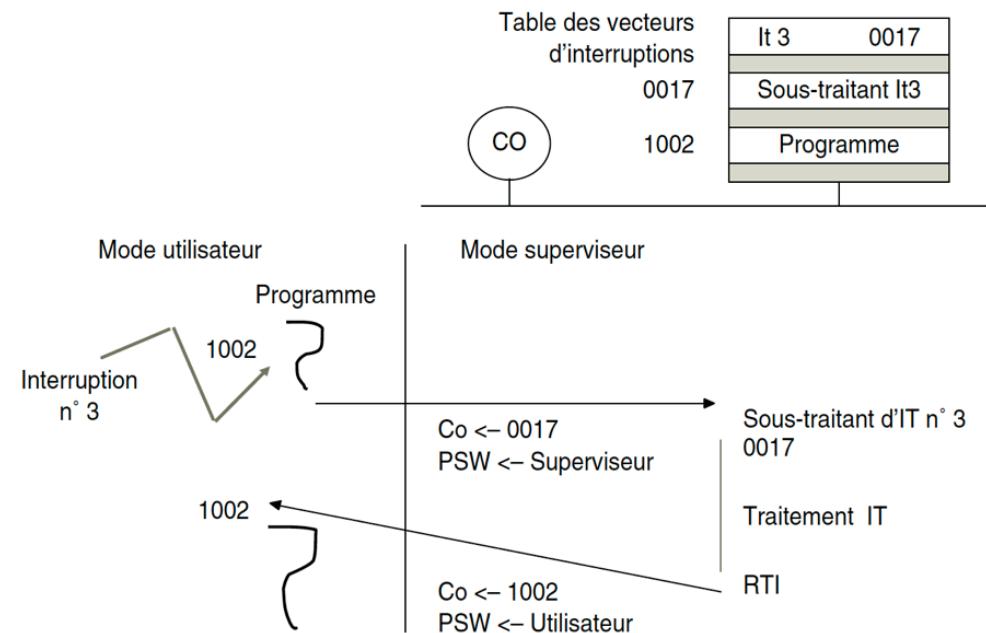
PRISE EN COMPTE D'UNE IRQ

- Enregistrer → **pile de l'OS**
 - l'adresse d'instruction interrompue
 - l'état du processeur (registres)



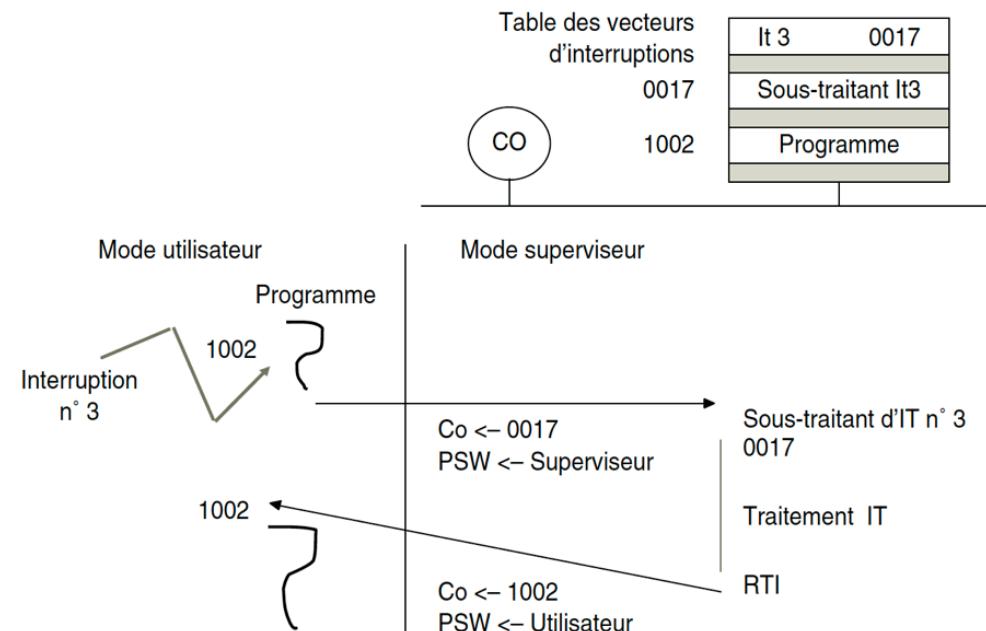
PRISE EN COMPTE D'UNE IRQ

- Enregistrer → **pile de l'OS**
 - l'adresse d'instruction interrompue
 - l'état du processeur (registres)
- Passer en **mode superviseur**



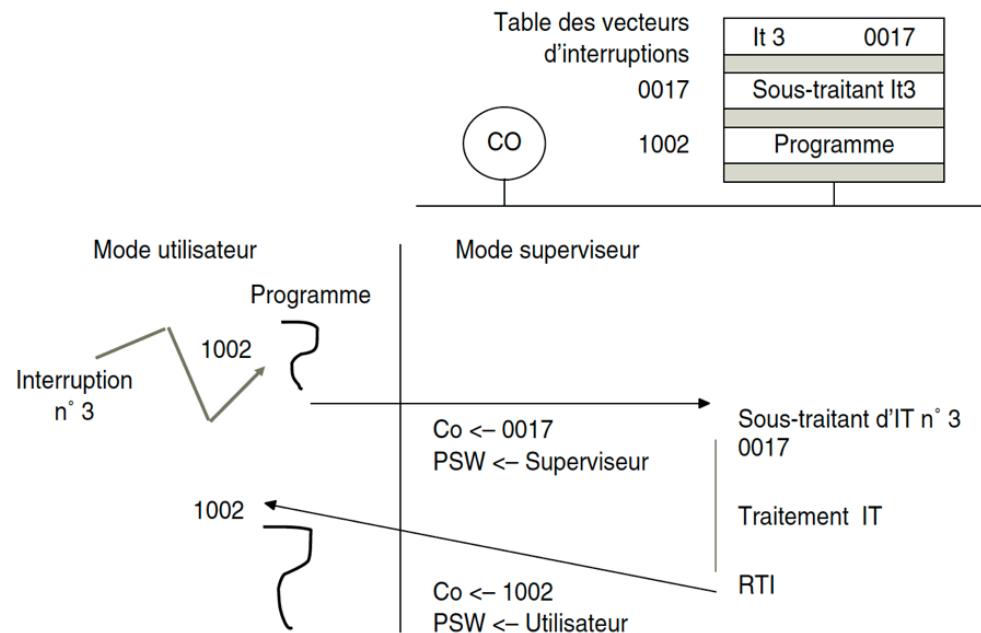
PRISE EN COMPTE D'UNE IRQ

- Enregistrer → **pile de l'OS**
 - l'adresse d'instruction interrompue
 - l'état du processeur (registres)
- Passer en **mode superviseur**
- Charger la **routine** correspondant à l'interruption

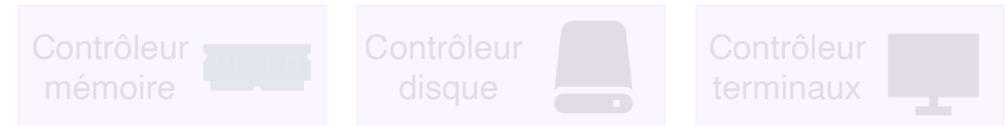


PRISE EN COMPTE D'UNE IRQ

- Enregistrer → **pile de l'OS**
 - l'adresse d'instruction interrompue
 - l'état du processeur (registres)
- Passer en **mode superviseur**
- Charger la **routine** correspondant à l'interruption
- **Contrôleur d'interruption**
👉 prioriser les interruptions



INTERFACE NOYAU-UTILISATEUR



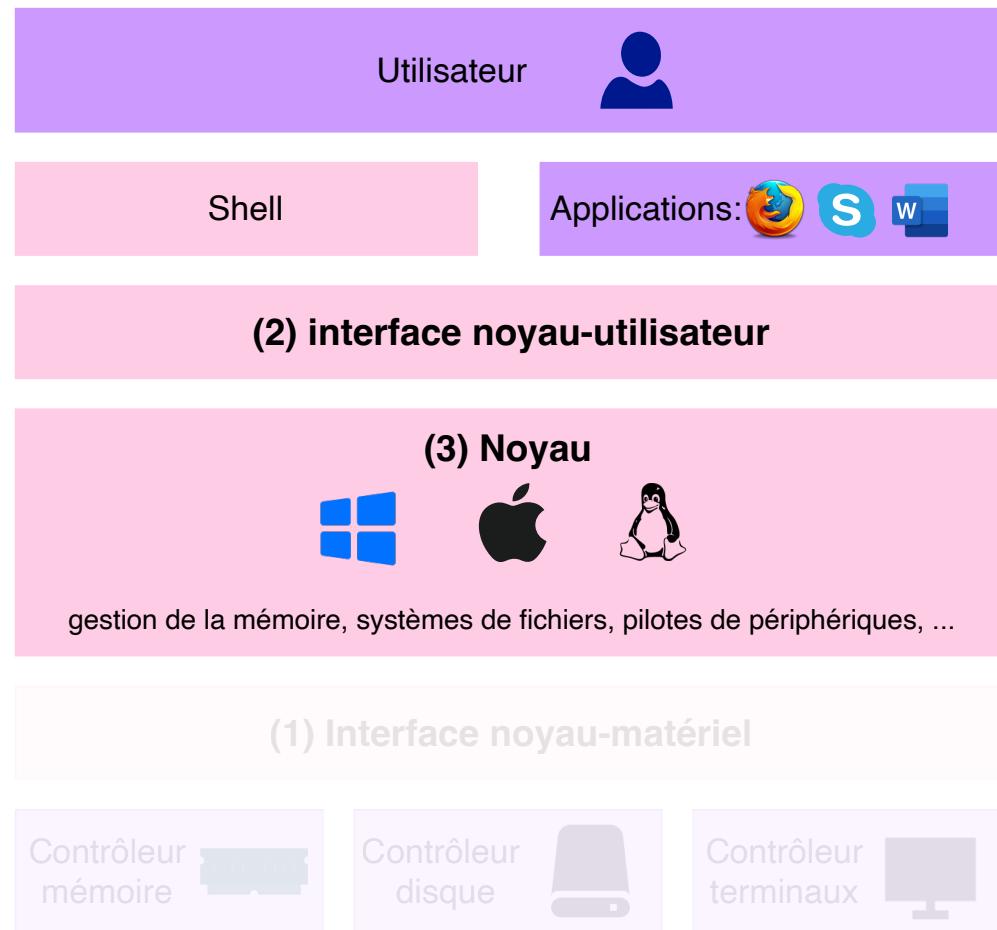
INTERFACE NOYAU-UTILISATEUR

- Présenter une **interface** entre le hardware et les applications.
👉 une **interface simplifiée et unifiée**.



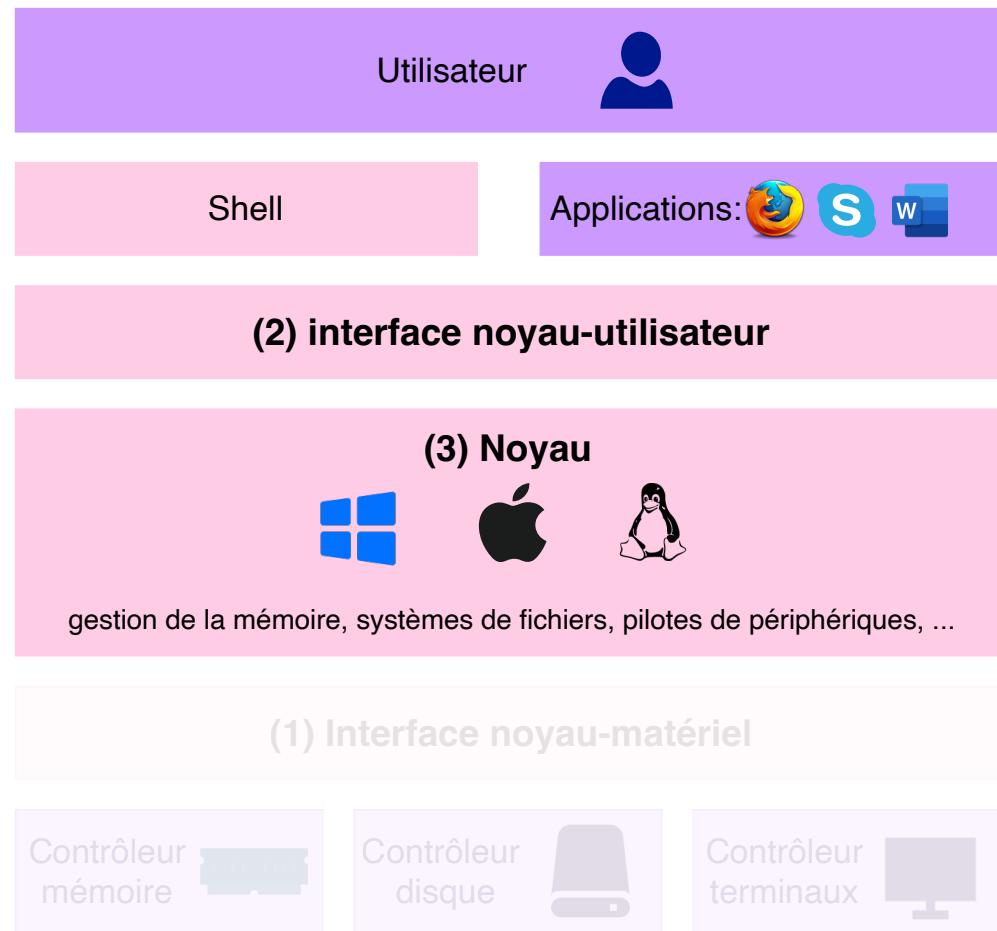
INTERFACE NOYAU-UTILISATEUR

- Présenter une **interface** entre le hardware et les applications.
👉 une **interface simplifiée et unifiée**.
- Présenter au-dessus de la machine physique, **une machine virtuelle** plus simple et plus conviviale.

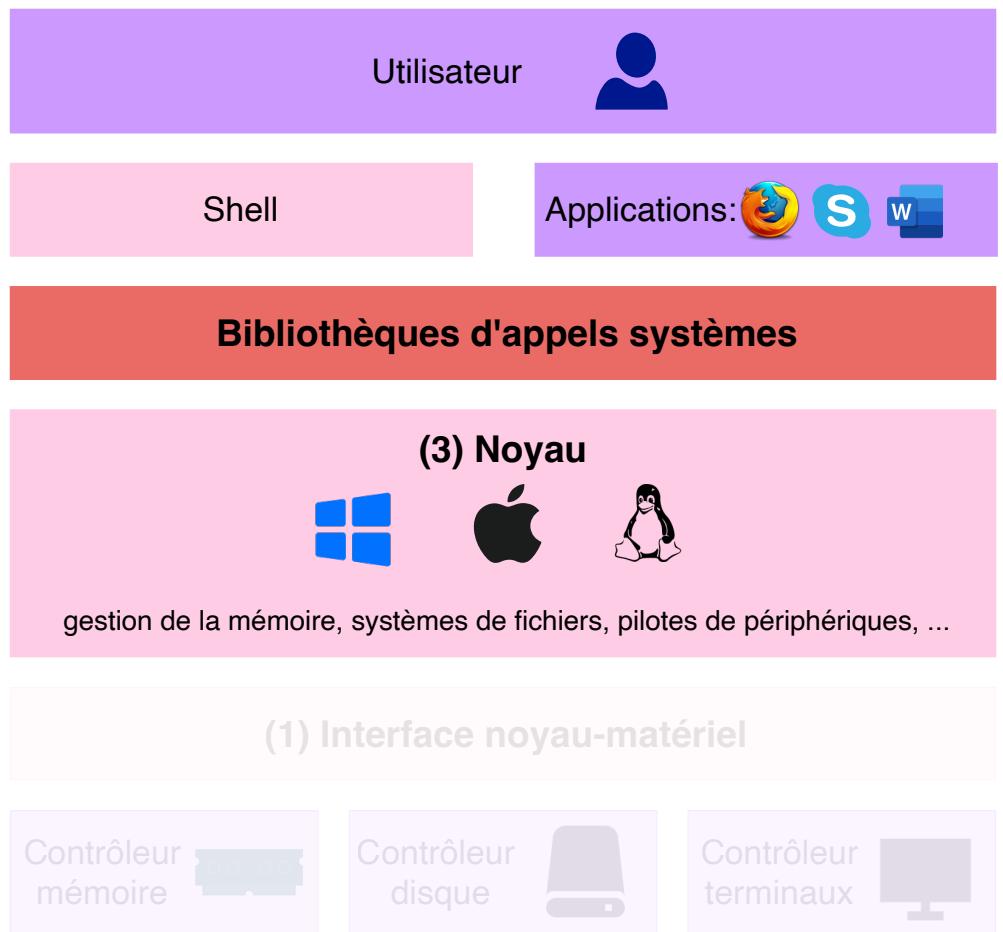


INTERFACE NOYAU-UTILISATEUR

- Présenter une **interface** entre le hardware et les applications.
👉 une **interface simplifiée et unifiée**.
- Présenter au-dessus de la machine physique, **une machine virtuelle** plus simple et plus conviviale.
- Créer **l'illusion de vrais ressources physiques** (processeur, mémoire, périphérique ...).

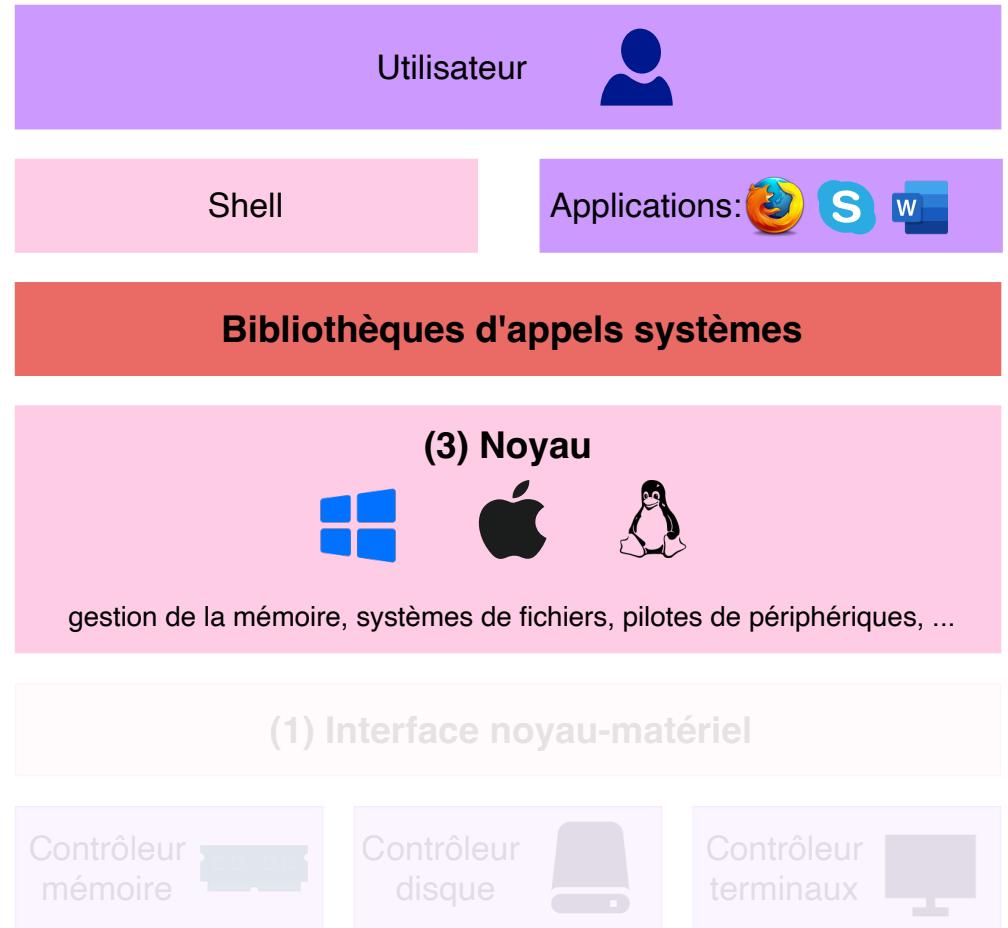


LES APPELS SYSTÈMES



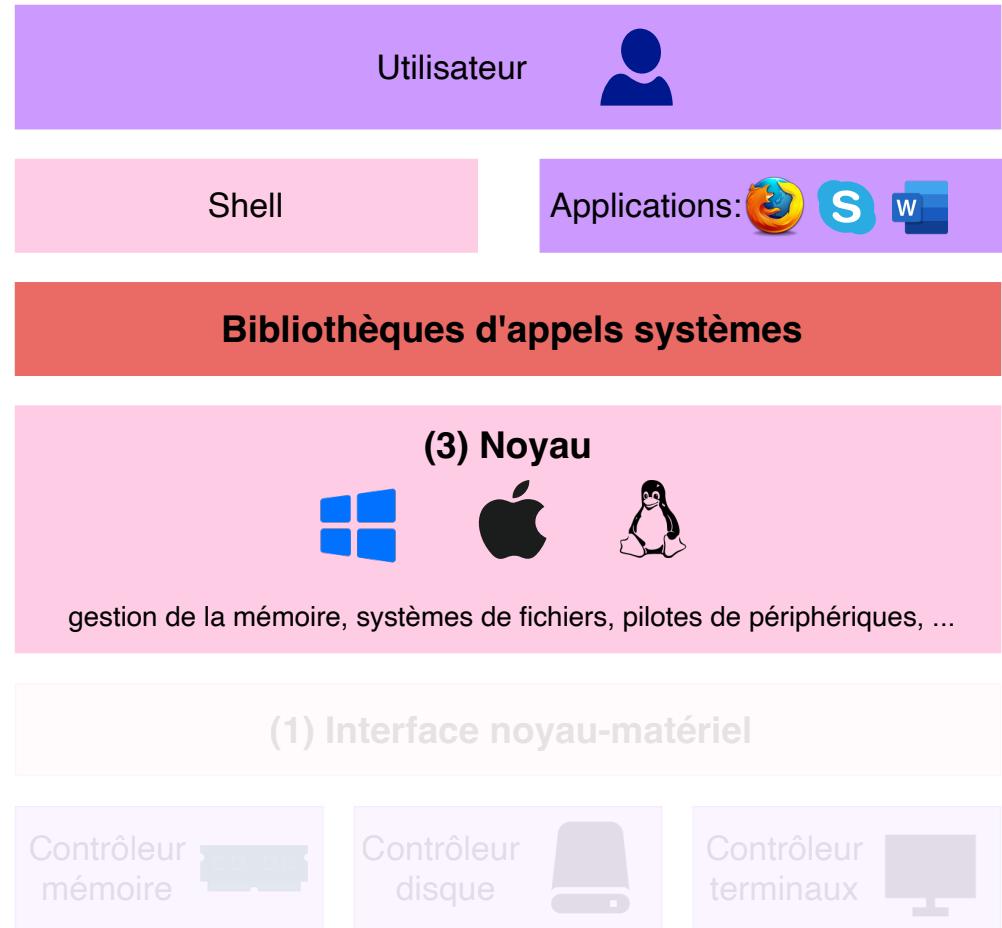
LES APPELS SYSTÈMES

- Fournir une **interface d'accès** aux ressources matérielles.



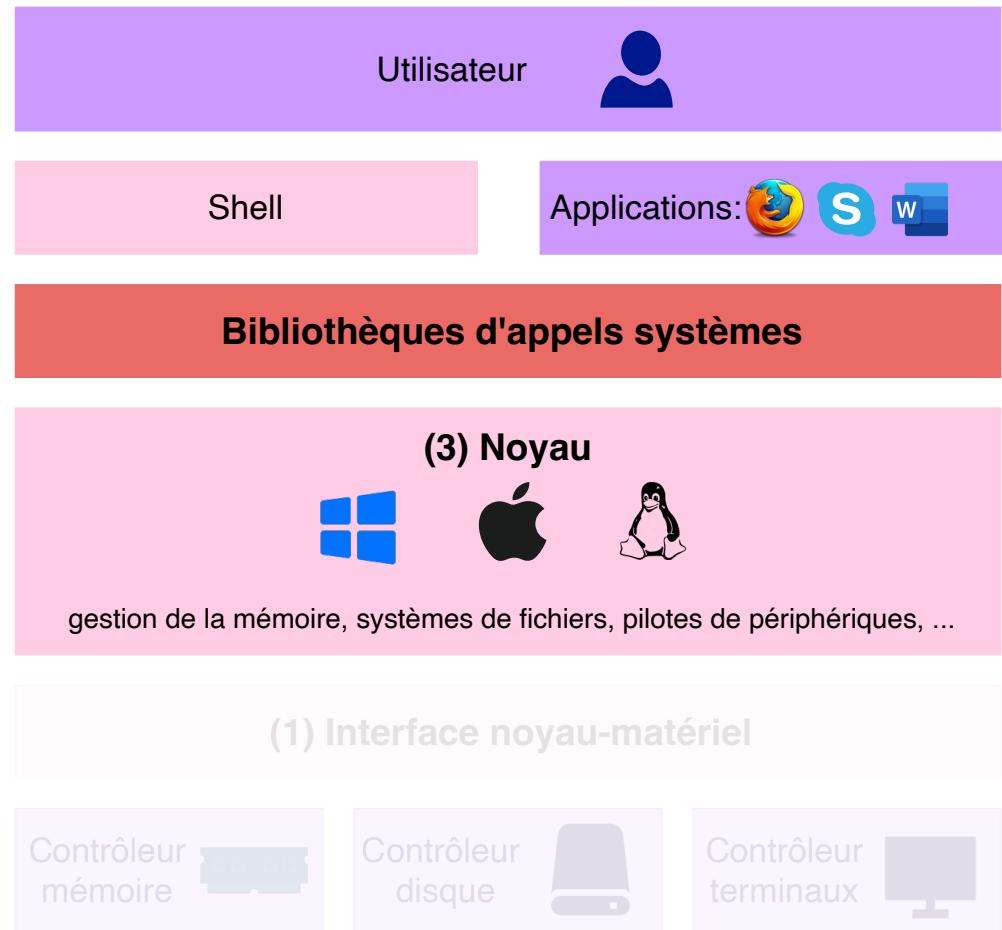
LES APPELS SYSTÈMES

- Fournir une **interface d'accès** aux ressources matérielles.
👉 par le biais de **fonctions prédéfinies** (**appels/routines systèmes**).



LES APPELS SYSTÈMES

- Fournir une **interface d'accès** aux ressources matérielles.
 - 👉 par le biais de **fonctions prédéfinies** (**appels/routines systèmes**).
 - 👉 les points d'entrées aux **fonctionnalités de l'OS**.



EXEMPLES D'APPELS SYSTÈMES

EXEMPLES D'APPELS SYSTÈMES

- Contrôle de processus

EXEMPLES D'APPELS SYSTÈMES

- Contrôle de processus
 - `sys_fork` : créer un processus

EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**
 - `sys_fork` : créer un processus
 - `sys_wait` : attendre la terminaison d'un processus

EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**
 - `sys_fork` : créer un processus
 - `sys_wait` : attendre la terminaison d'un processus
 - `sys_exit` : terminer l'exécution d'un processus

EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**
 - `sys_fork` : créer un processus
 - `sys_wait` : attendre la terminaison d'un processus
 - `sys_exit` : terminer l'exécution d'un processus
 - `sys_kill` : Envoyer un signal à un processus

EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**
 - `sys_fork` : créer un processus
 - `sys_wait` : attendre la terminaison d'un processus
 - `sys_exit` : terminer l'exécution d'un processus
 - `sys_kill` : Envoyer un signal à un processus
- **Gestion des fichiers**

EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**

- `sys_fork` : créer un processus
- `sys_wait` : attendre la terminaison d'un processus
- `sys_exit` : terminer l'exécution d'un processus
- `sys_kill` : Envoyer un signal à un processus

- **Gestion des fichiers**

- `sys_open/sys_close` : ouvrir/fermer un fichier

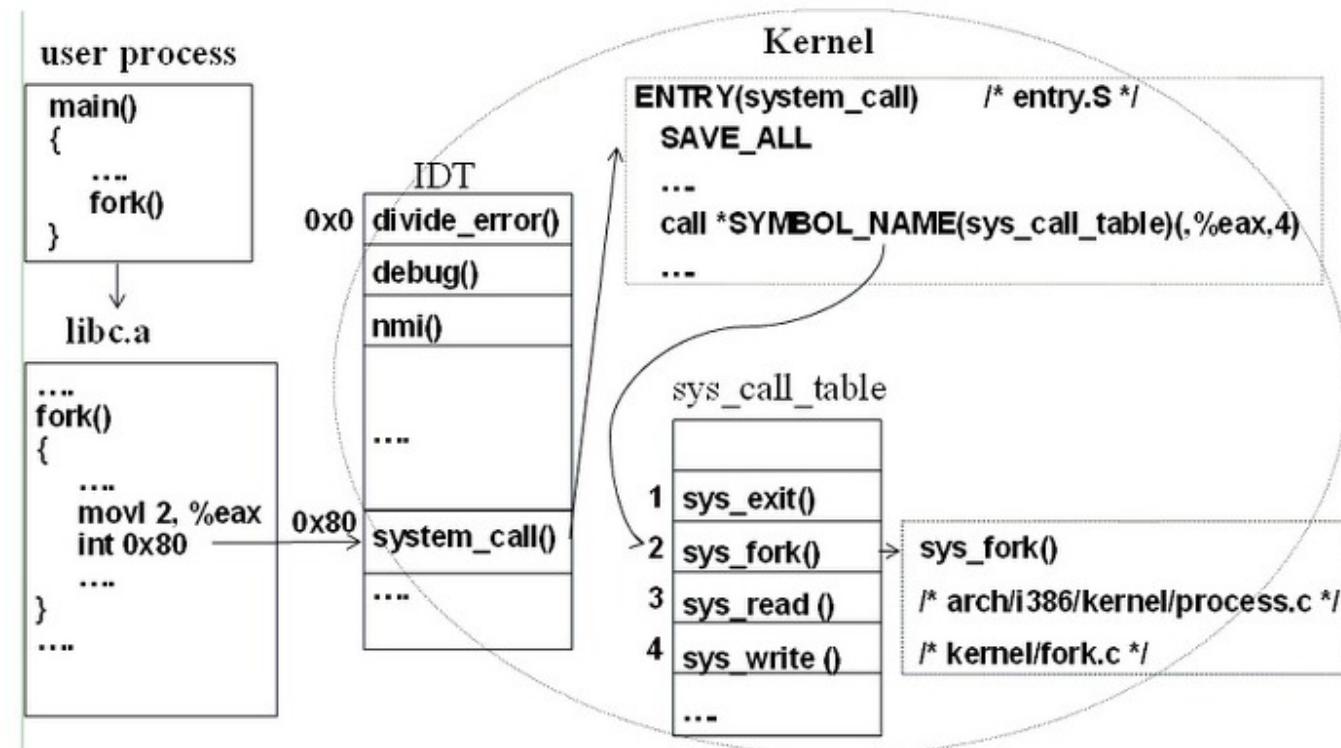
EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**
 - `sys_fork` : créer un processus
 - `sys_wait` : attendre la terminaison d'un processus
 - `sys_exit` : terminer l'exécution d'un processus
 - `sys_kill` : Envoyer un signal à un processus
- **Gestion des fichiers**
 - `sys_open/sys_close` : ouvrir/fermer un fichier
 - `sys_read/sys_write` : lire/écrire des données dans un fichier

EXEMPLES D'APPELS SYSTÈMES

- **Contrôle de processus**
 - `sys_fork` : créer un processus
 - `sys_wait` : attendre la terminaison d'un processus
 - `sys_exit` : terminer l'exécution d'un processus
 - `sys_kill` : Envoyer un signal à un processus
- **Gestion des fichiers**
 - `sys_open/sys_close` : ouvrir/fermer un fichier
 - `sys_read/sys_write` : lire/écrire des données dans un fichier
 - `sys_mkdir/sys_rmdir` : créer/supprimer un répertoire

L'APPEL SYSTÈME `fork()`



EXEMPLE SOUS UNIX

EXAMPLE SOUS UNIX

- L'instruction `os.chdir(path)` permet de changer le répertoire courant d'un programme **Python** en cours d'exécution.

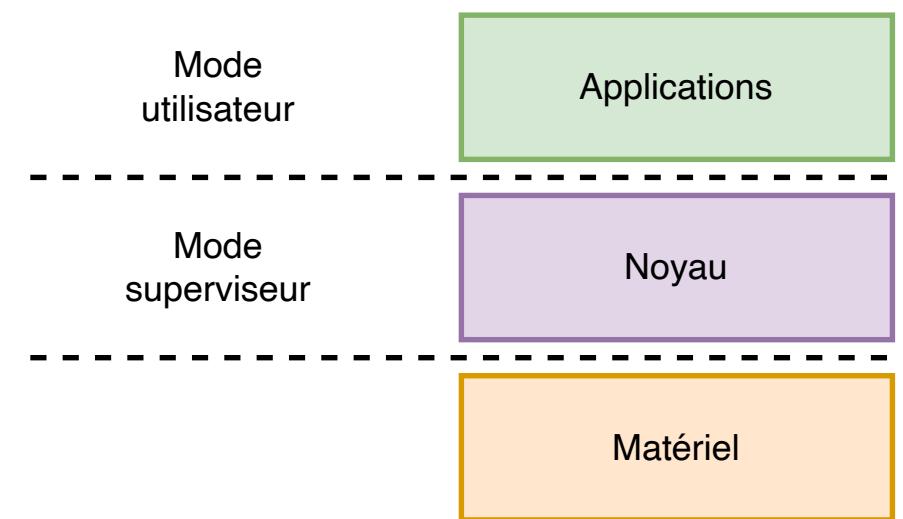
EXAMPLE SOUS UNIX

- L'instruction `os.chdir(path)` permet de changer le répertoire courant d'un programme **Python** en cours d'exécution.
- La commande `cd path` permet de changer le répertoire courant depuis l'**interpréteur de commandes (Shell)**.

EXAMPLE SOUS UNIX

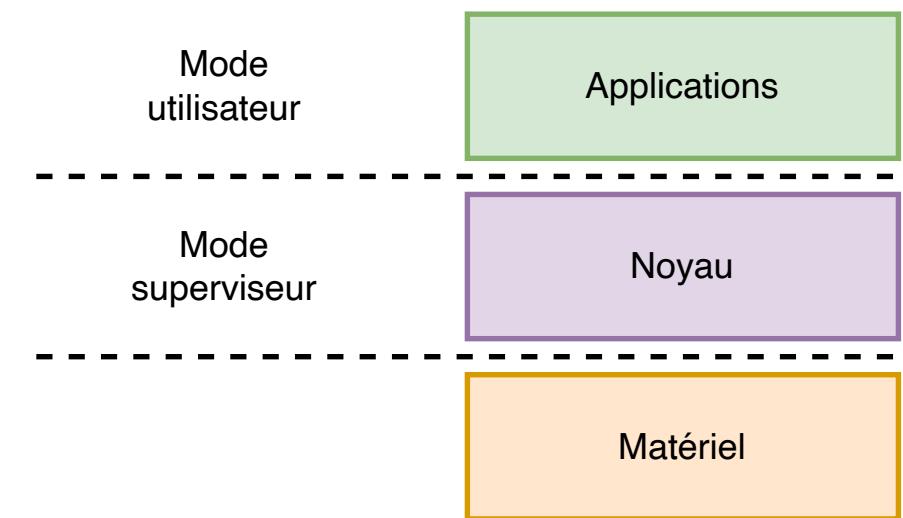
- L'instruction `os.chdir(path)` permet de changer le répertoire courant d'un programme **Python** en cours d'exécution.
- La commande `cd path` permet de changer le répertoire courant depuis l'**interpréteur de commandes (Shell)**.
- Les deux exécutent la routine système `sys_chdir`.

MODES D'EXÉCUTIONS



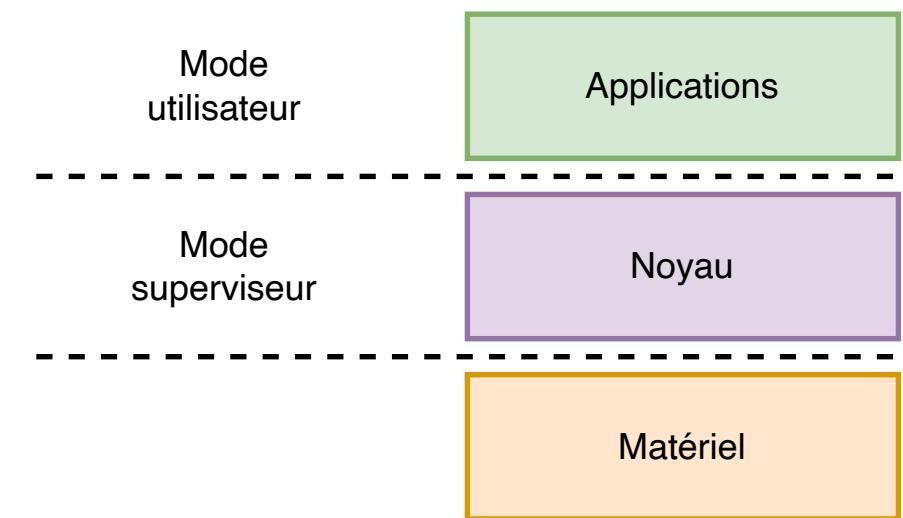
MODES D'EXÉCUTIONS

- **Un programme utilisateur** s'exécute dans un **mode utilisateur** :



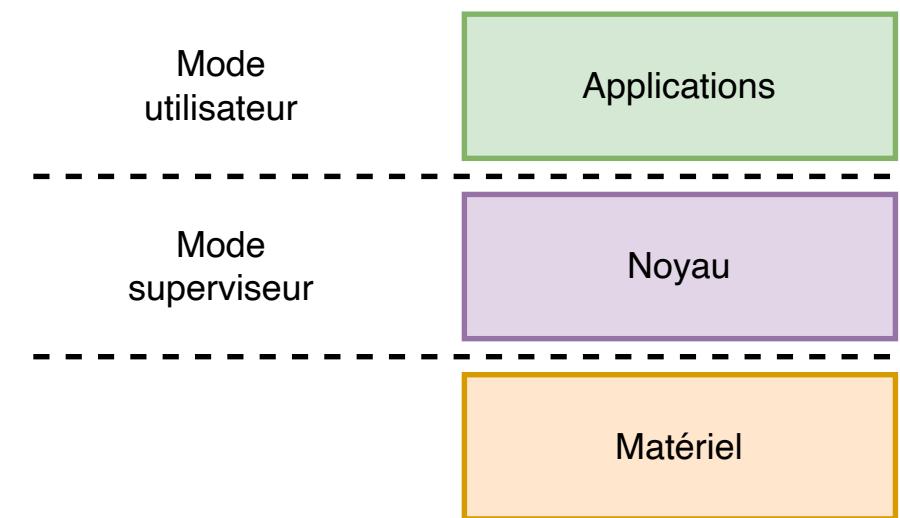
MODES D'EXÉCUTIONS

- **Un programme utilisateur** s'exécute dans un **mode utilisateur** :
 - Un Jeu d'**instructions restreint** pour protéger la machine.



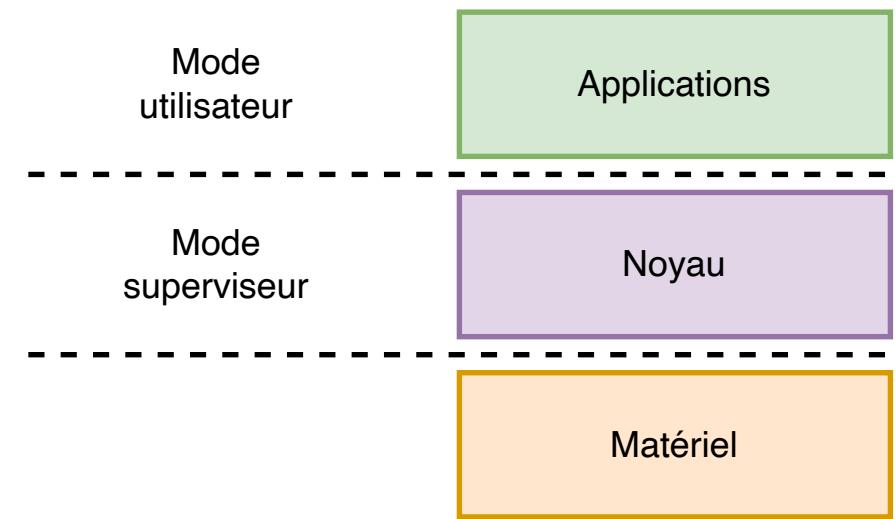
MODES D'EXÉCUTIONS

- **Un programme utilisateur** s'exécute dans un **mode utilisateur** :
 - Un Jeu d'**instructions restreint** pour protéger la machine.
 - **ex.** manipulation des IRQs interdite.



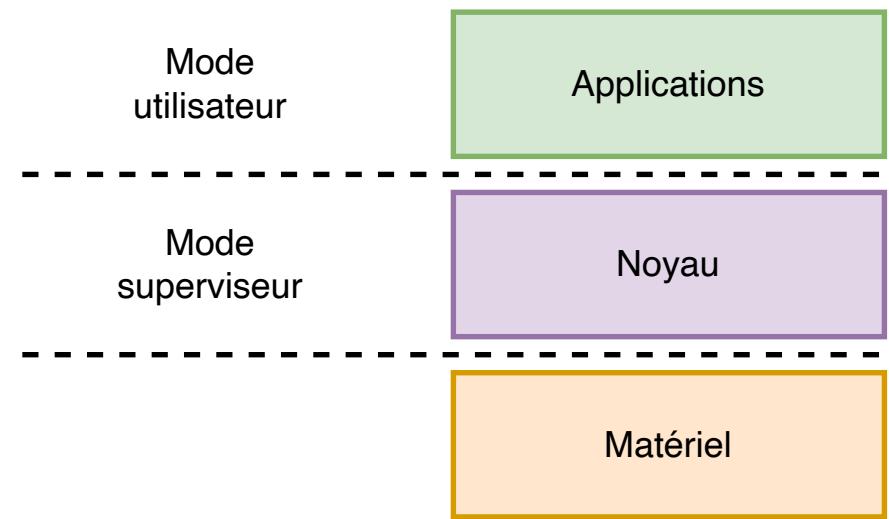
MODES D'EXÉCUTIONS

- **Un programme utilisateur** s'exécute dans un **mode utilisateur** :
 - Un Jeu d'**instructions restreint** pour protéger la machine.
 - **ex.** manipulation des IRQs interdite.
- L'**OS** s'exécute dans **un mode privilégié (mode superviseur)**:

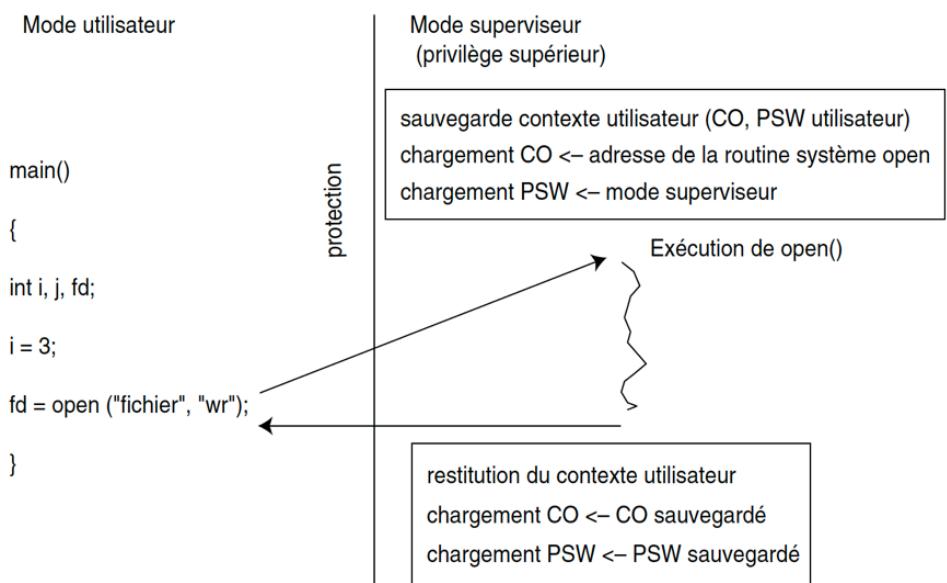


MODES D'EXÉCUTIONS

- **Un programme utilisateur** s'exécute dans un **mode utilisateur** :
 - Un Jeu d'**instructions restreint** pour protéger la machine.
 - **ex.** manipulation des IRQs interdite.
- L'**OS** s'exécute dans un **mode privilégié (mode superviseur)**:
 - aucune restriction de droits n'existe.

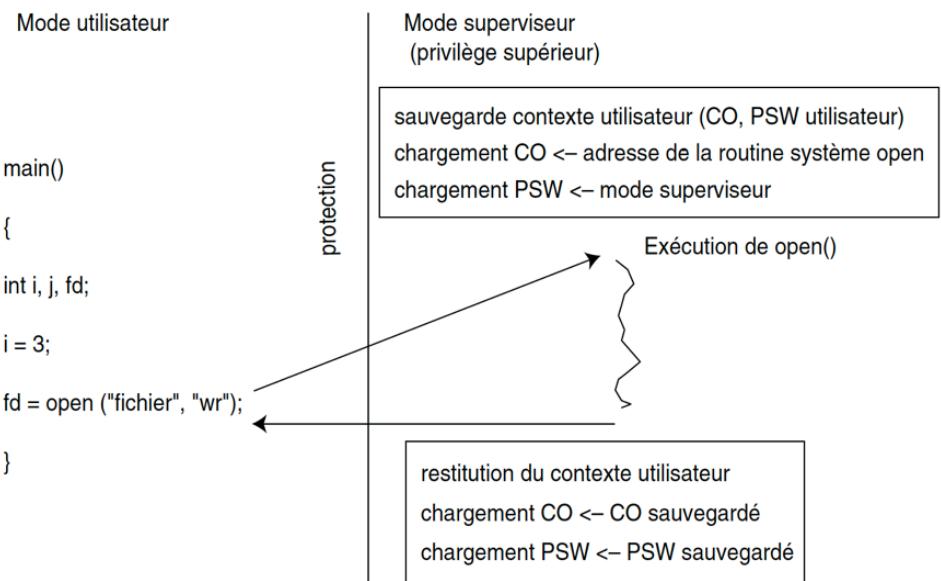


COMMUTATIONS DE CONTEXTE



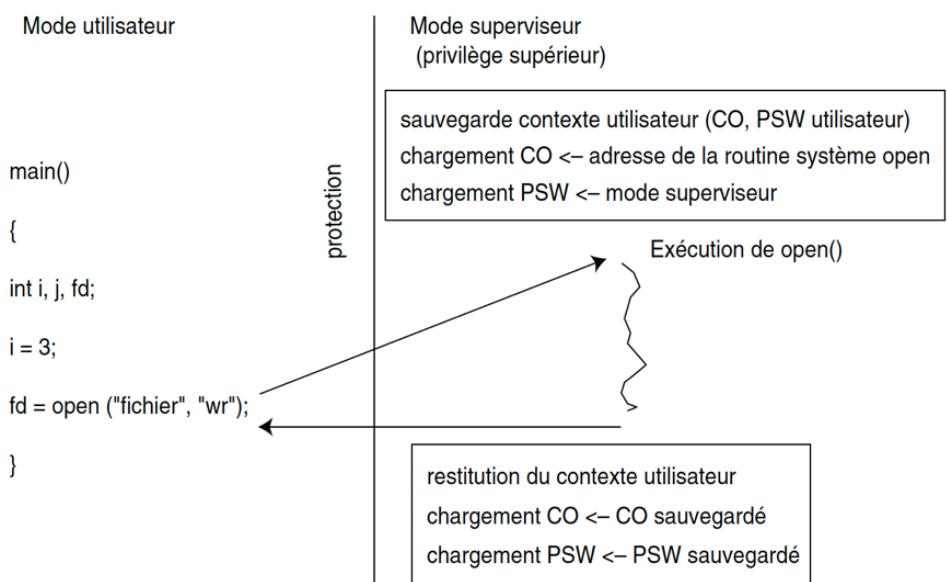
COMMUTATIONS DE CONTEXTE

- A l'appel d'une fonction du noyau,
il y a passage au **mode superviseur**
(commutation de contexte).



COMMUTATIONS DE CONTEXTE

- A l'appel d'une fonction du noyau, il y a passage au **mode superviseur** (commutation de contexte).
- A la fin de l'exécution de la fonction du noyau, le programme repasse au **mode utilisateur**.

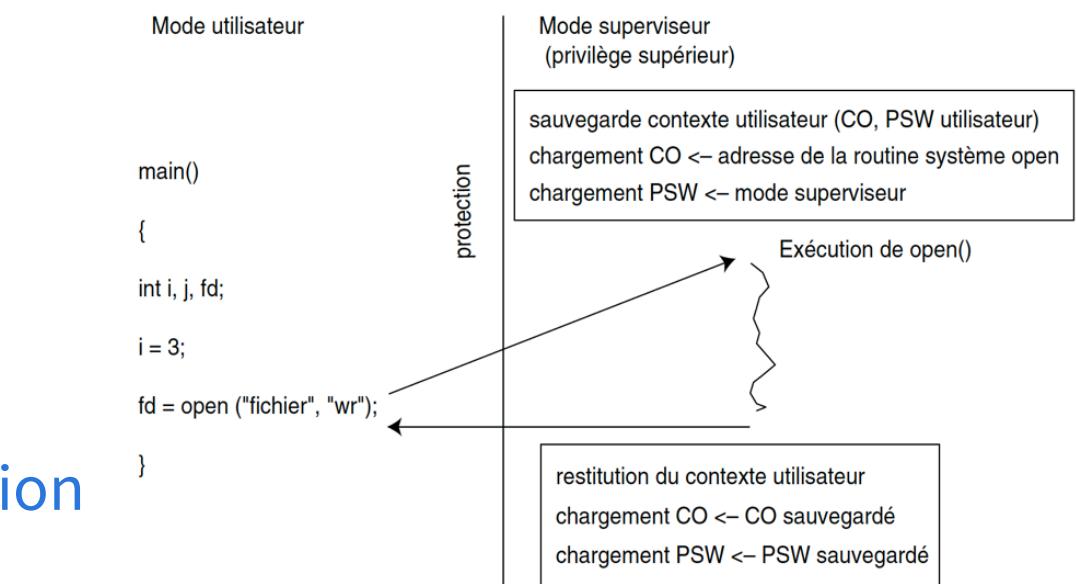


COMMUTATIONS DE CONTEXTE

- A l'appel d'une fonction du noyau, il y a passage au **mode superviseur** (commutation de contexte).

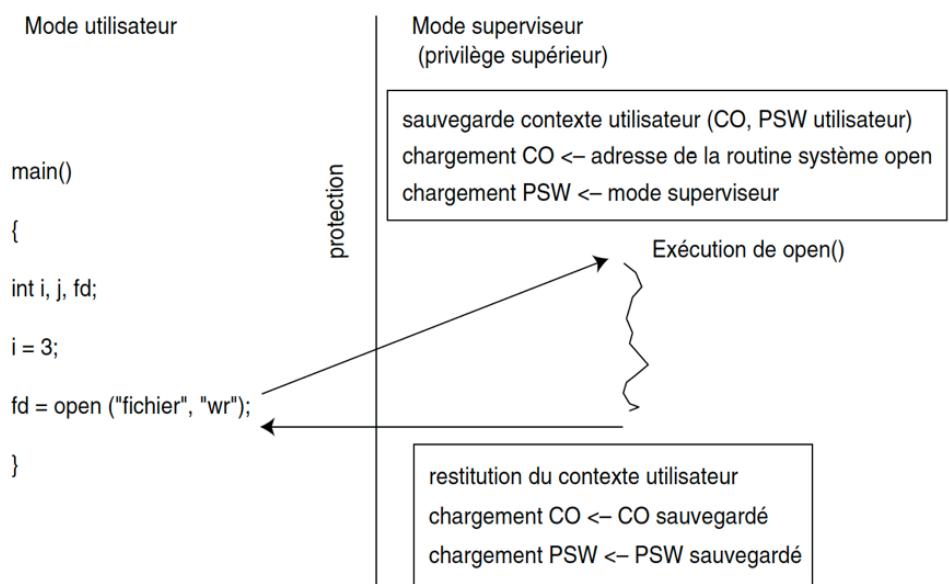
- A la fin de l'exécution de la fonction du noyau, le programme repasse au **mode utilisateur**.

- commutation de contexte avec restauration du contexte utilisateur.

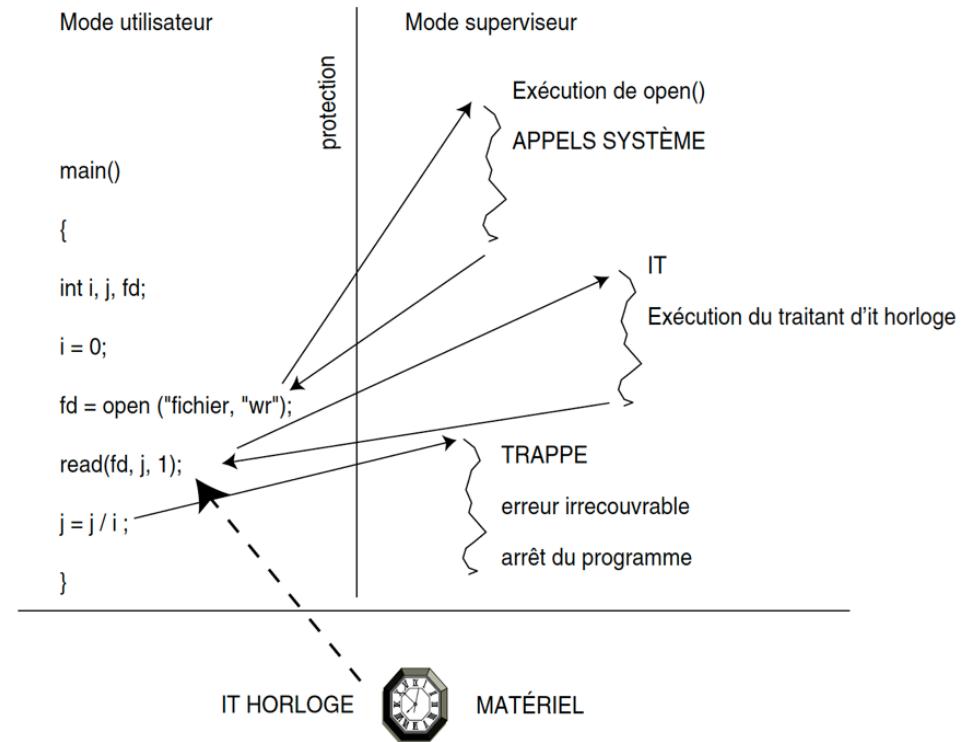


COMMUTATIONS DE CONTEXTE

- A l'appel d'une fonction du noyau, il y a passage au **mode superviseur** (commutation de contexte).
- A la fin de l'exécution de la fonction du noyau, le programme repasse au **mode utilisateur**.
 - commutation de contexte avec **restauration du contexte utilisateur**.
 - reprise de l'exécution du programme utilisateur

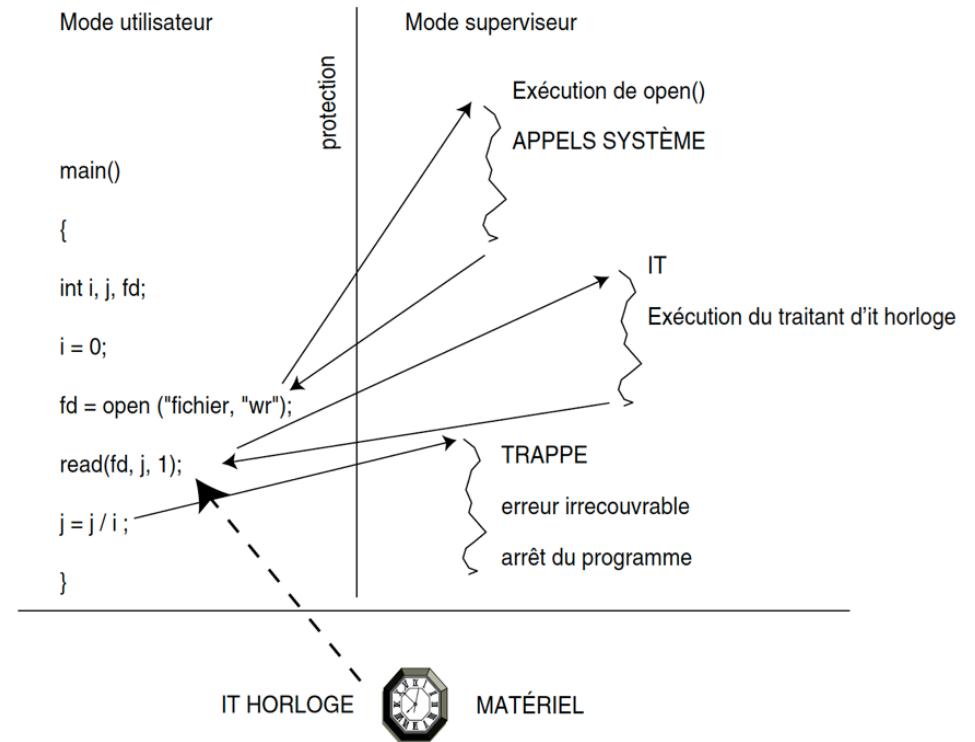


LES CAUSES DE COMMUTATIONS DE CONTEXTE



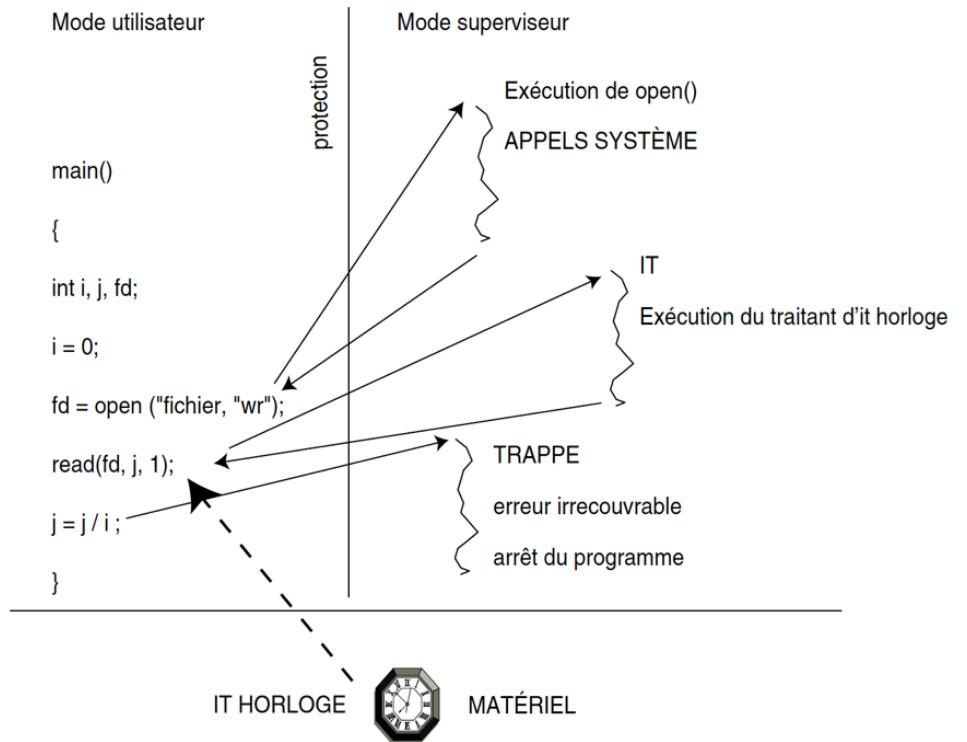
LES CAUSES DE COMMUTATIONS DE CONTEXTE

1. appelle d'une **fonction système**.



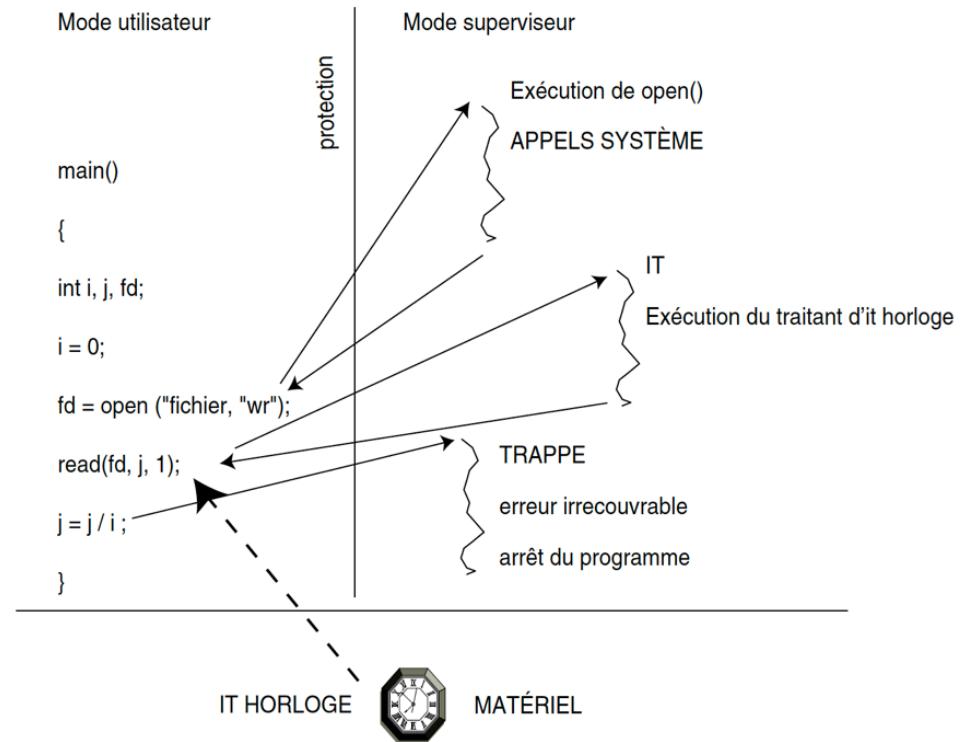
LES CAUSES DE COMMUTATIONS DE CONTEXTE

1. appelle d'une **fonction système**.
2. exécute une opération illicite
(trappe ou exception).



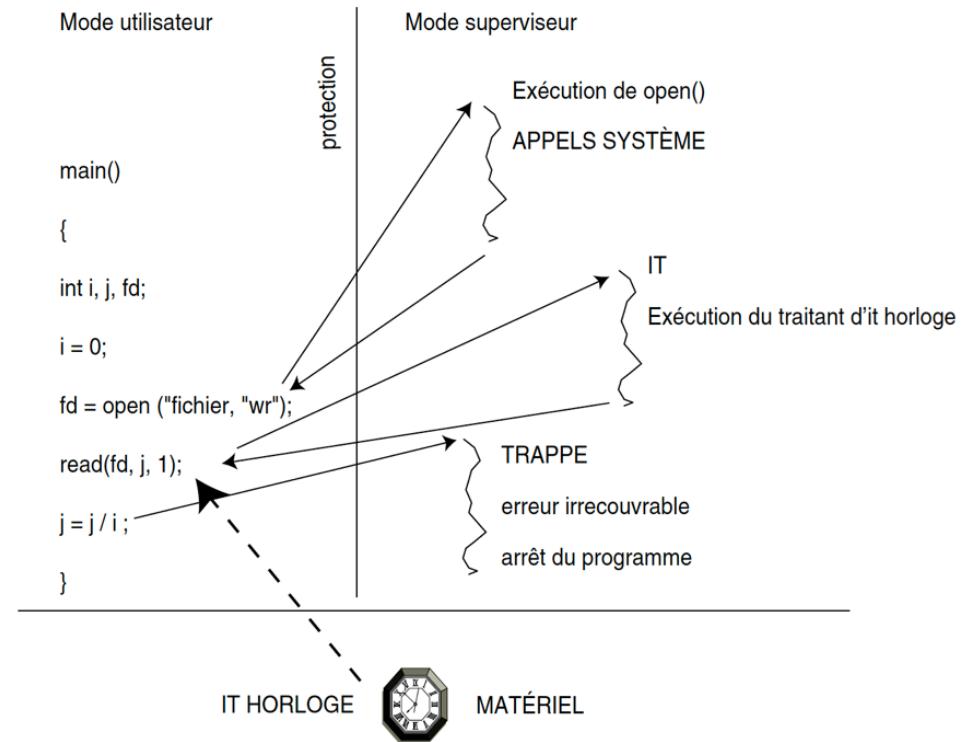
LES CAUSES DE COMMUTATIONS DE CONTEXTE

1. appelle d'une **fonction système**.
2. exécute une opération illicite (**trappe** ou **exception**).
3. prise en compte d'une **interruption matérielle**.



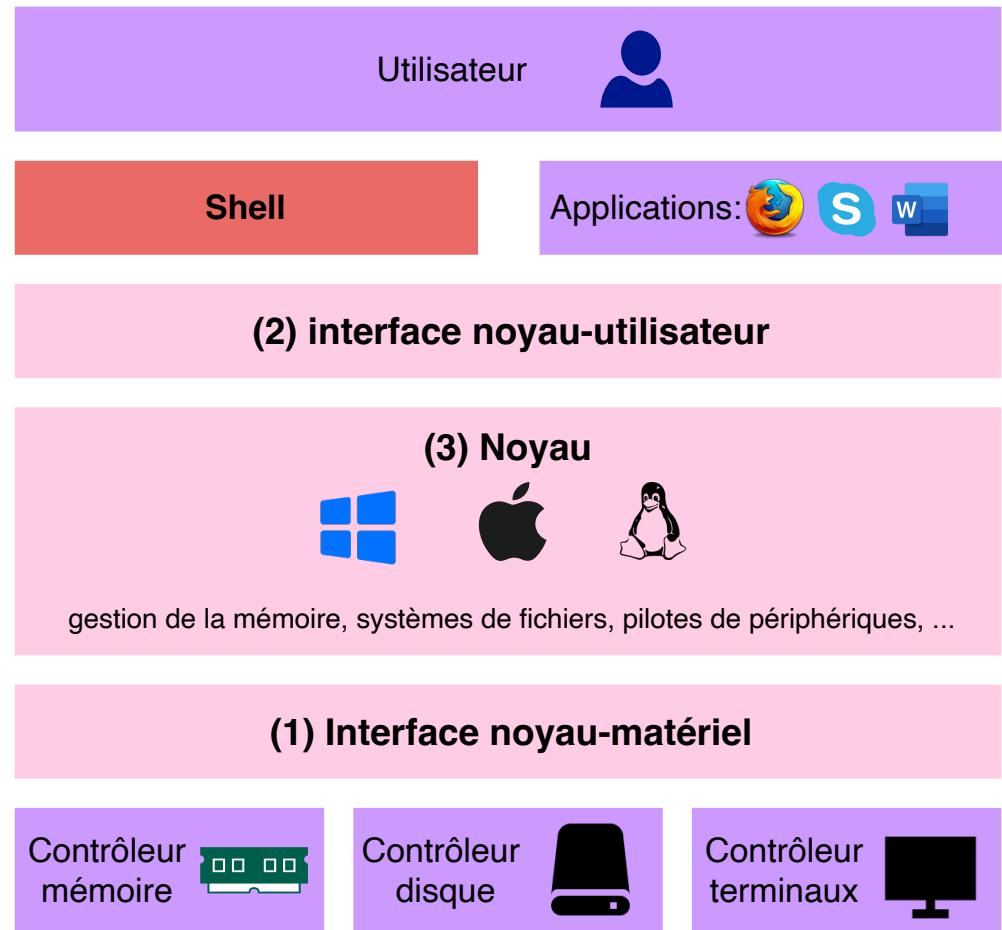
LES CAUSES DE COMMUTATIONS DE CONTEXTE

1. appelle d'une **fonction système**.
2. exécute une opération illicite (**trappe** ou **exception**).
3. prise en compte d'une **interruption matérielle**.



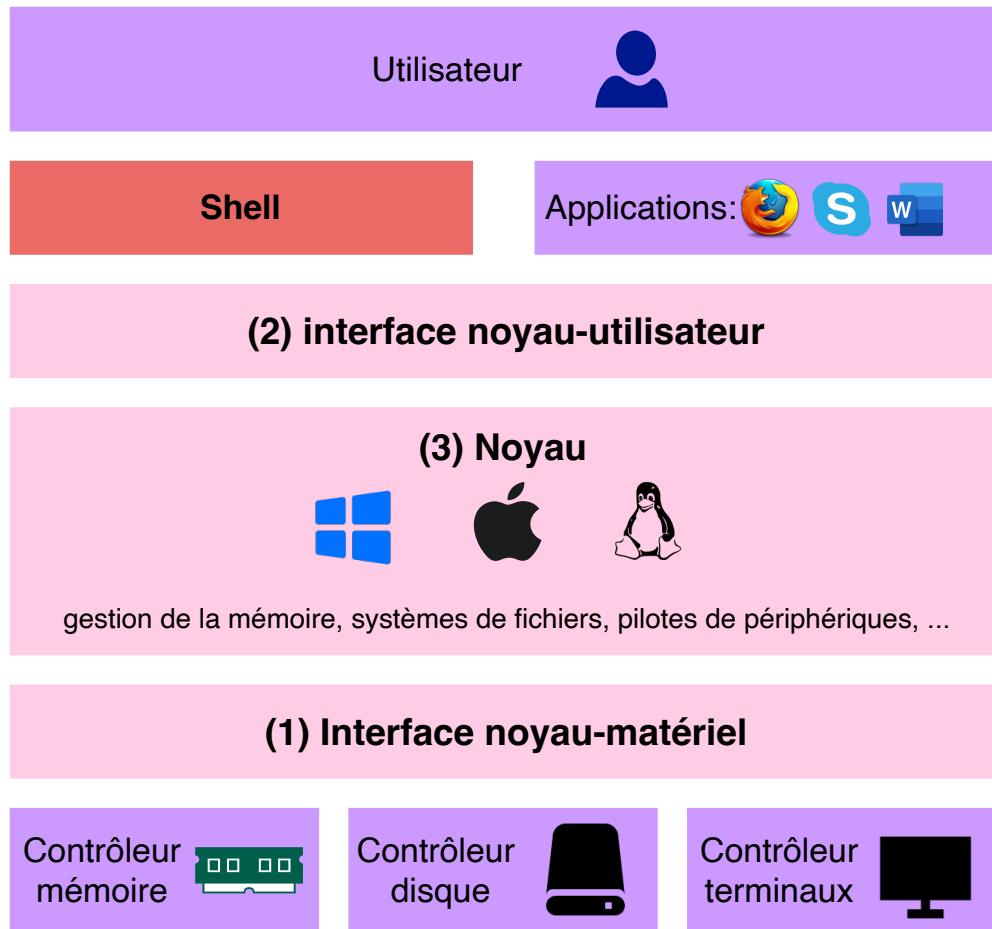
Trappes et appels systèmes sont parfois qualifiés **d'interruptions logicielles** par opposition aux interruptions matérielles.

INTERPRÉTEUR DE COMMANDE (SHELL)



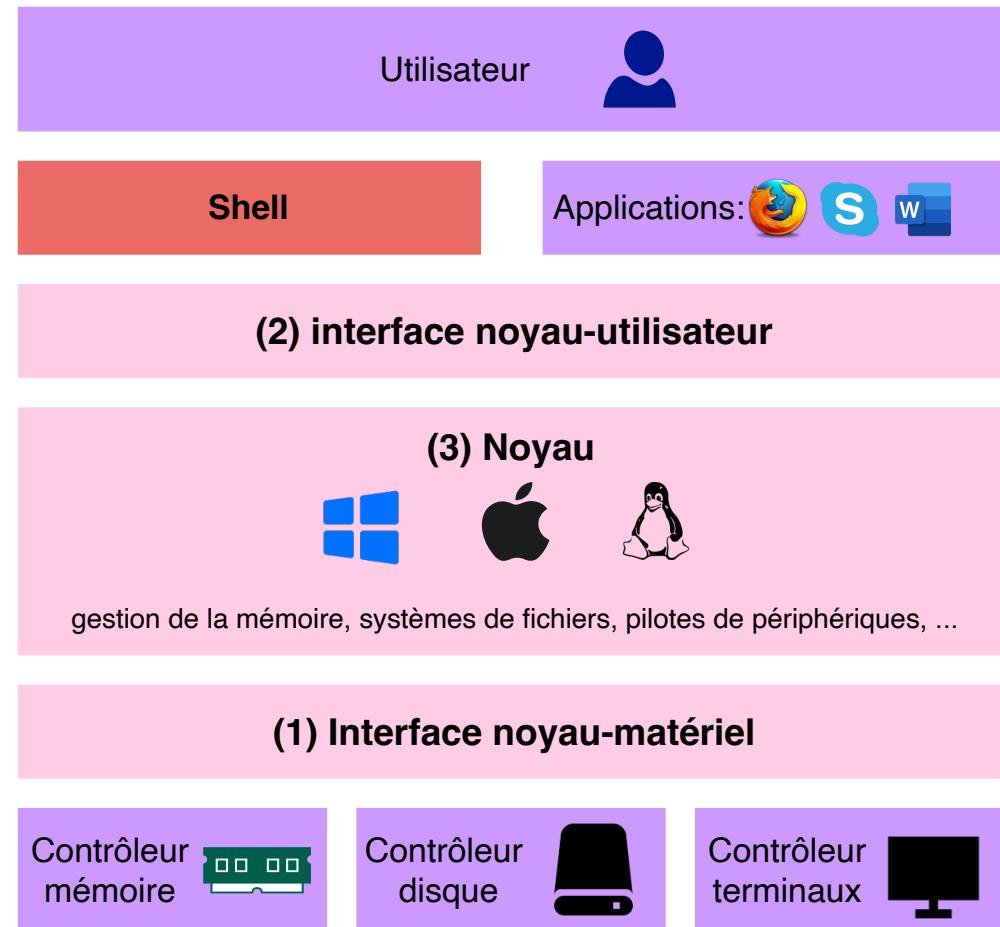
INTERPRÉTEUR DE COMMANDE (SHELL)

- **Langage de commandes :**
l'interface de niveau utilisateur
avec le système d'exploitation.



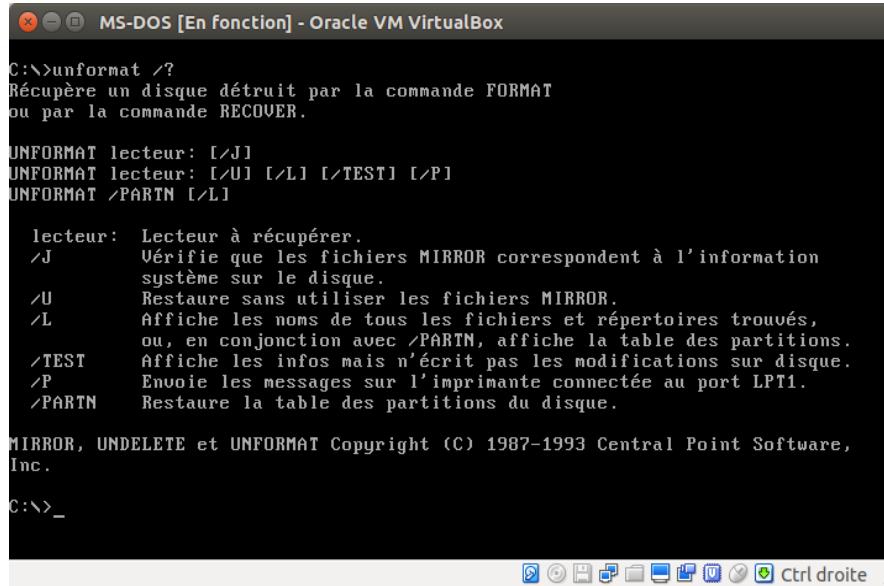
INTERPRÉTEUR DE COMMANDE (SHELL)

- **Langage de commandes** : l'interface de niveau utilisateur avec le système d'exploitation.
- **Interpréteur de commandes** : exécuter des commandes de l'utilisateur en appellant la routine système appropriée.



INTERPRÉTEUR DE COMMANDE (SHELL)

- Chaque système d'exploitation a son propre **langage de commandes** :



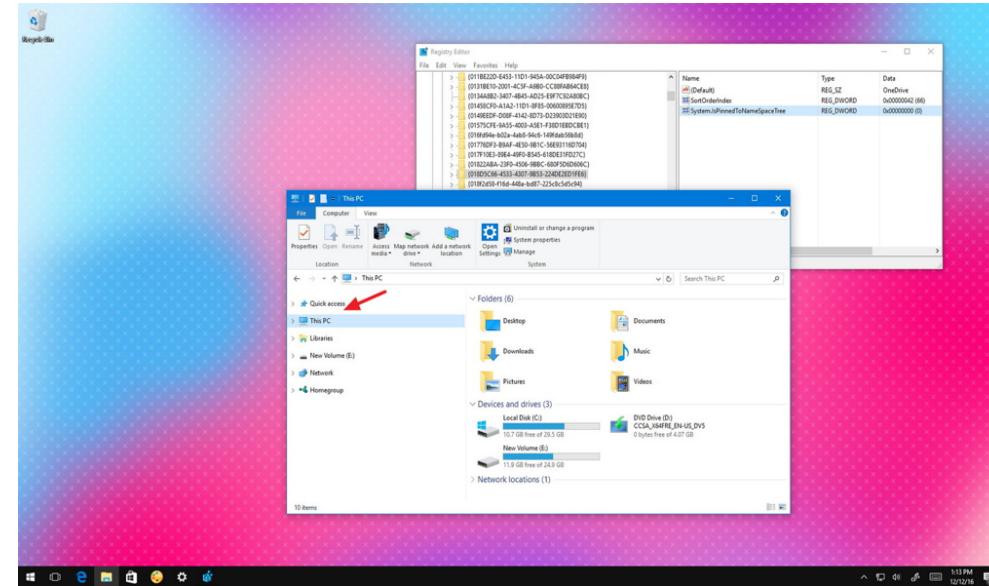
```
C:\>unformat /?
Récupère un disque détruit par la commande FORMAT
ou par la commande RECOVER.

UNFORMAT lecteur: [/J]
UNFORMAT lecteur: [/U] [/L] [/TEST] [/P]
UNFORMAT [/PARTN [/L]

lecteur: Lecteur à récupérer.
/J      Vérifie que les fichiers MIRROR correspondent à l'information
       système sur le disque.
/U      Restaure sans utiliser les fichiers MIRROR.
/L      Affiche les noms de tous les fichiers et répertoires trouvés,
       ou, en conjonction avec /PARTN, affiche la table des partitions.
/TEST   Affiche les infos mais n'écrit pas les modifications sur disque.
/P      Envoie les messages sur l'imprimante connectée au port LPT1.
/PARTN  Restaure la table des partitions du disque.

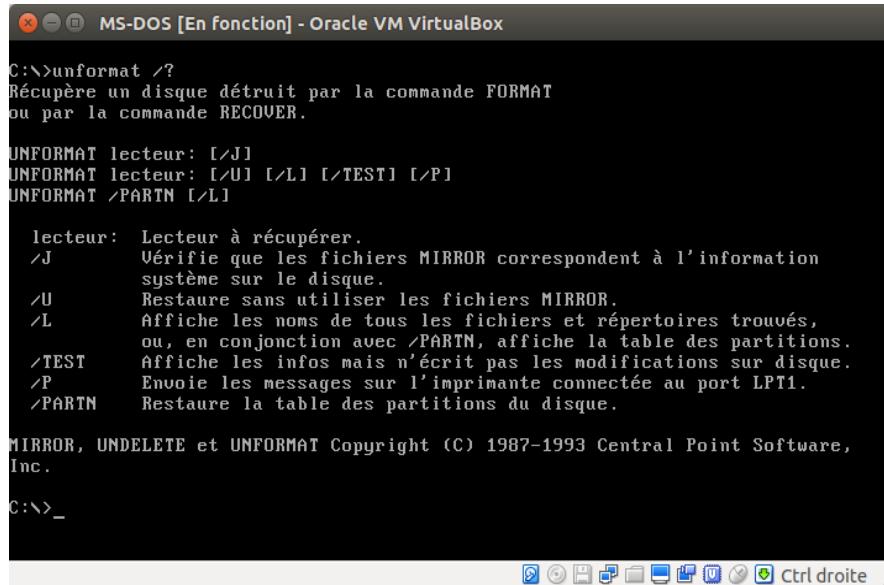
MIRROR, UNDELETE et UNFORMAT Copyright (C) 1987-1993 Central Point Software,
Inc.

C:\>_
```



INTERPRÉTEUR DE COMMANDE (SHELL)

- Chaque système d'exploitation a son propre **langage de commandes** :
 - **MSDOS/Unix** : console + clavier



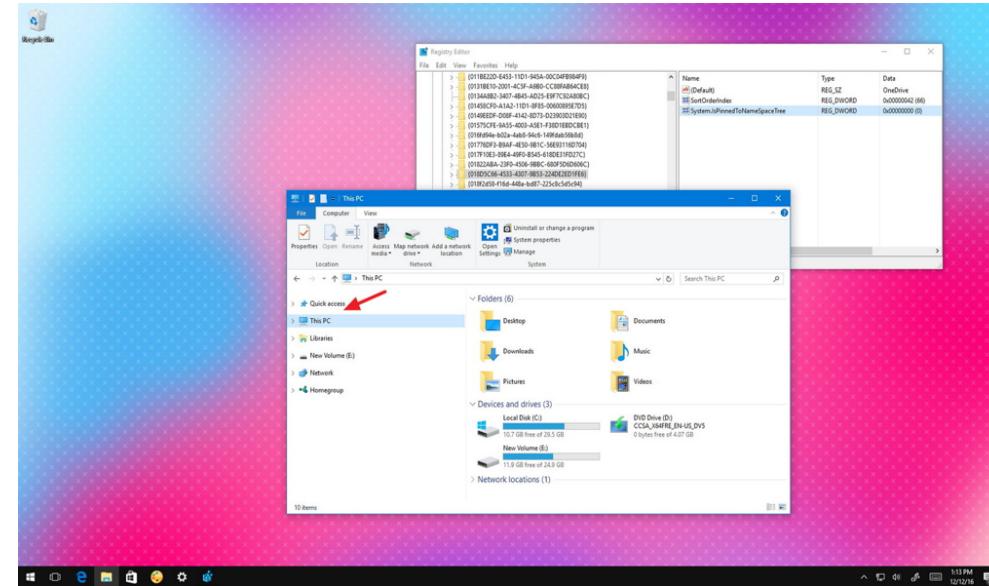
```
C:\>unformat /?
Récupère un disque détruit par la commande FORMAT
ou par la commande RECOVER.

UNFORMAT lecteur: [/J]
UNFORMAT lecteur: [/U] [/L] [/TEST] [/P]
UNFORMAT [/PARTN [/L]

lecteur: Lecteur à récupérer.
/J    Vérifie que les fichiers MIRROR correspondent à l'information
      système sur le disque.
/U    Restaure sans utiliser les fichiers MIRROR.
/L    Affiche les noms de tous les fichiers et répertoires trouvés,
      ou, en conjonction avec /PARTN, affiche la table des partitions.
/TEST   Affiche les infos mais n'écrit pas les modifications sur disque.
/P    Envoie les messages sur l'imprimante connectée au port LPT1.
/PARTN  Restaure la table des partitions du disque.

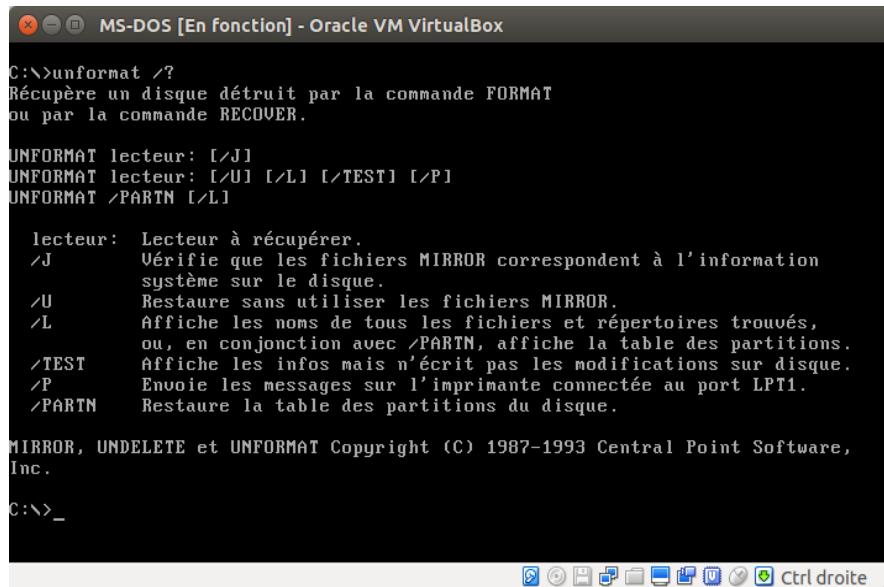
MIRROR, UNDELETE et UNFORMAT Copyright (C) 1987-1993 Central Point Software,
Inc.

C:\>_
```



INTERPRÉTEUR DE COMMANDE (SHELL)

- Chaque système d'exploitation a son propre **langage de commandes** :
 - **MSDOS/Unix** : console + clavier
 - **Mac/Windows** : souris + clavier



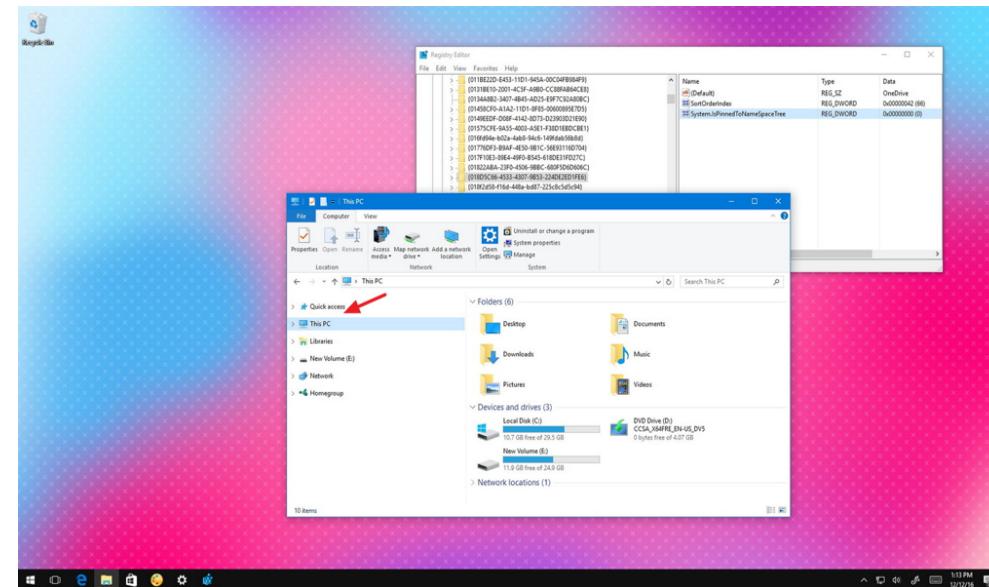
```
C:\>unformat /?
Récupère un disque détruit par la commande FORMAT
ou par la commande RECOVER.

UNFORMAT lecteur: [/J]
UNFORMAT lecteur: [/U] [/L] [/TEST] [/P]
UNFORMAT [/PARTN [/L]

lecteur: Lecteur à récupérer.
/J      Vérifie que les fichiers MIRROR correspondent à l'information
       système sur le disque.
/U      Restaure sans utiliser les fichiers MIRROR.
/L      Affiche les noms de tous les fichiers et répertoires trouvés,
       ou, en conjonction avec /PARTN, affiche la table des partitions.
/TEST   Affiche les infos mais n'écrit pas les modifications sur disque.
/P      Envoie les messages sur l'imprimante connectée au port LPT1.
/PARTN  Restaure la table des partitions du disque.

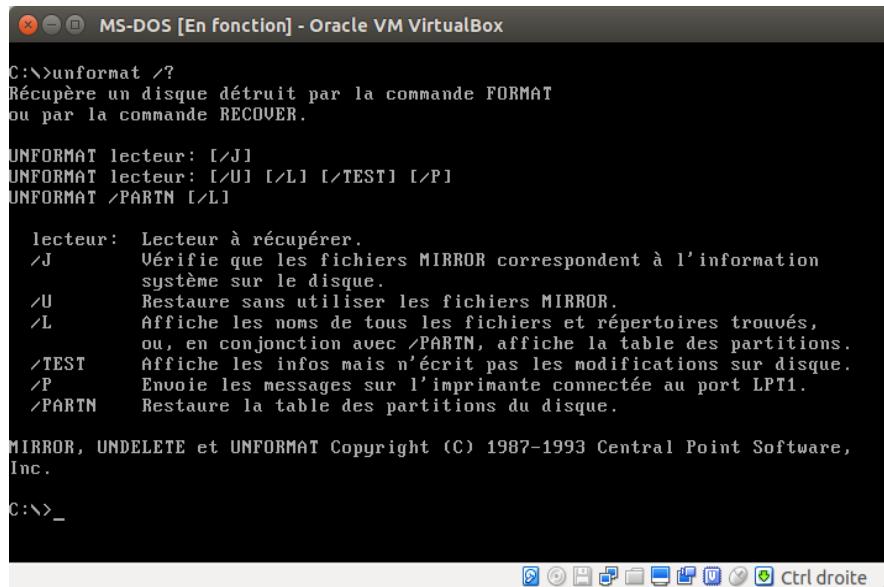
MIRROR, UNDELETE et UNFORMAT Copyright (C) 1987-1993 Central Point Software,
Inc.

C:\>_
```



INTERPRÉTEUR DE COMMANDE (SHELL)

- Chaque système d'exploitation a son propre **langage de commandes** :
 - **MSDOS/Unix** : console + clavier
 - **Mac/Windows** : souris + clavier
 - **iOS/Android** : boutons + écran tactile



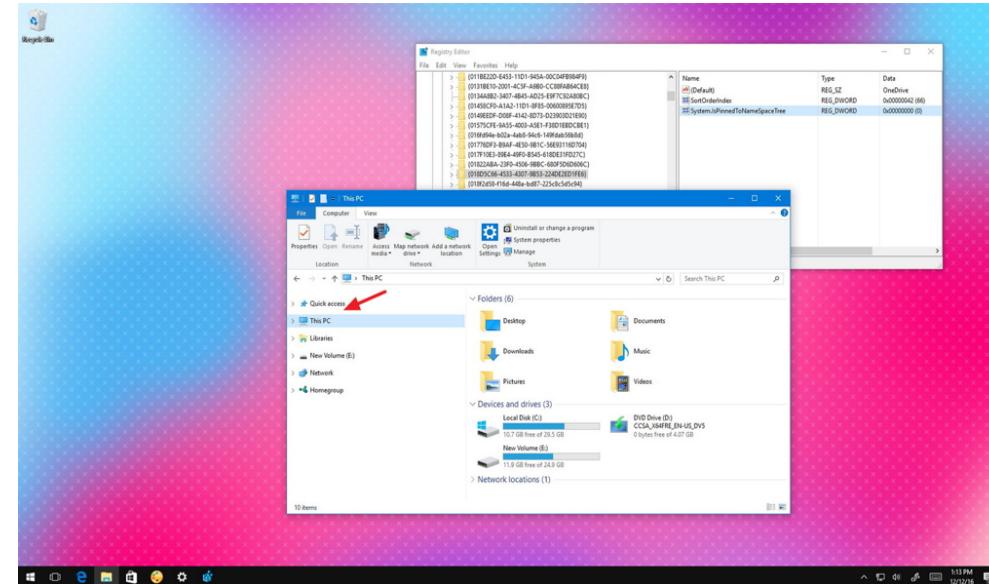
```
C:\>unformat /?
Récupère un disque détruit par la commande FORMAT
ou par la commande RECOVER.

UNFORMAT lecteur: [/J]
UNFORMAT lecteur: [/U] [/L] [/TEST] [/P]
UNFORMAT [/PARTN [/L]

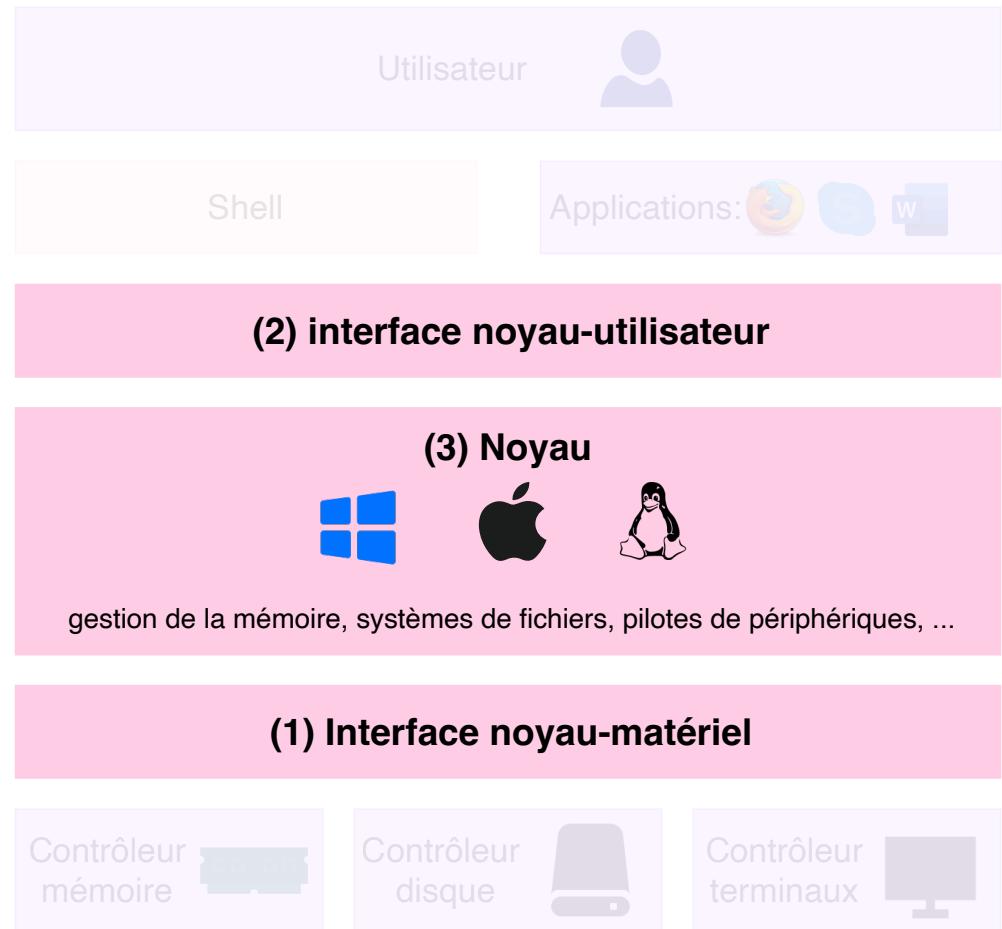
lecteur: Lecteur à récupérer.
/J    Vérifie que les fichiers MIRROR correspondent à l'information
      système sur le disque.
/U    Restaure sans utiliser les fichiers MIRROR.
/L    Affiche les noms de tous les fichiers et répertoires trouvés,
      ou, en conjonction avec /PARTN, affiche la table des partitions.
/TEST  Affiche les infos mais n'écrit pas les modifications sur disque.
/P    Envoie les messages sur l'imprimante connectée au port LPT1.
/PARTN Restaure la table des partitions du disque.

MIRROR, UNDELETE et UNFORMAT Copyright (C) 1987-1993 Central Point Software,
Inc.

C:\>_
```



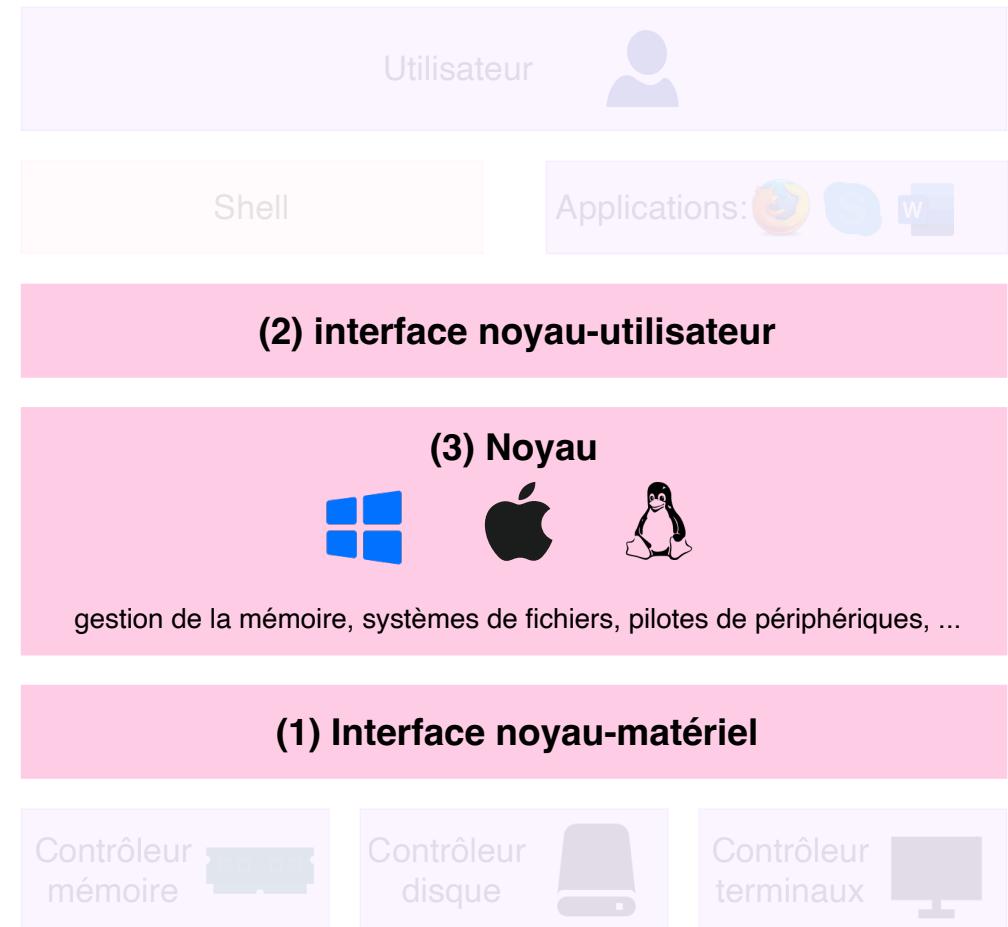
NOYAU D'UN SYSTÈME D'EXPLOITATION



NOYAU D'UN SYSTÈME D'EXPLOITATION

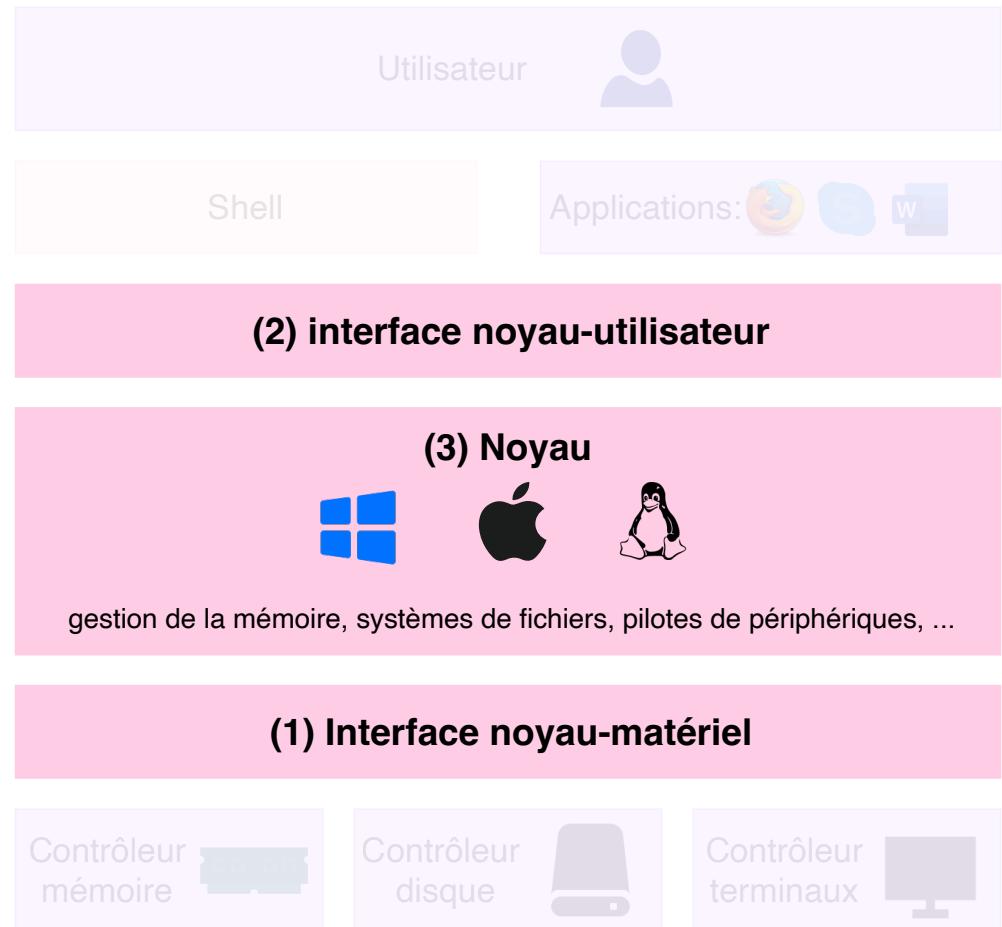
- **Gestion des entrées/sorties (I/O)**

- contrôleurs, pilotes, ...



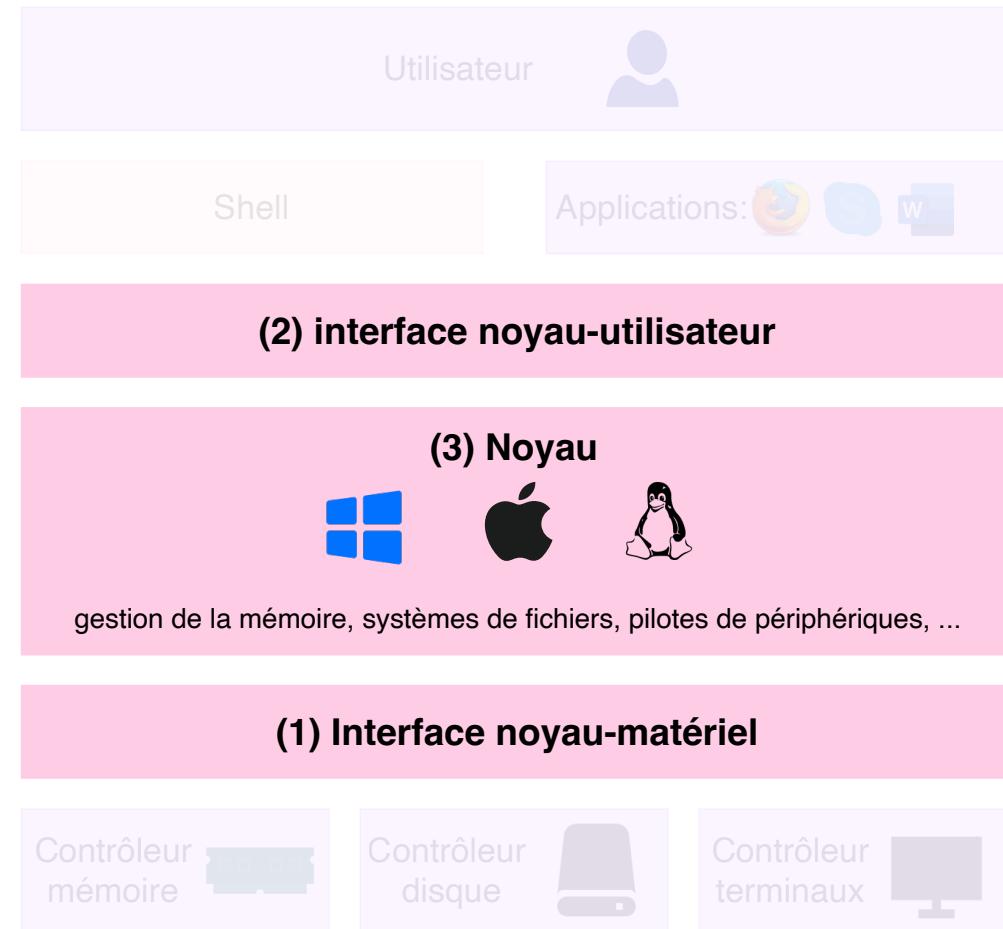
NOYAU D'UN SYSTÈME D'EXPLOITATION

- **Gestion des entrées/sorties (I/O)**
 - contrôleurs, pilotes, ...
- **Gestion des processus**
 - ordonnancement, synchronisation, ...



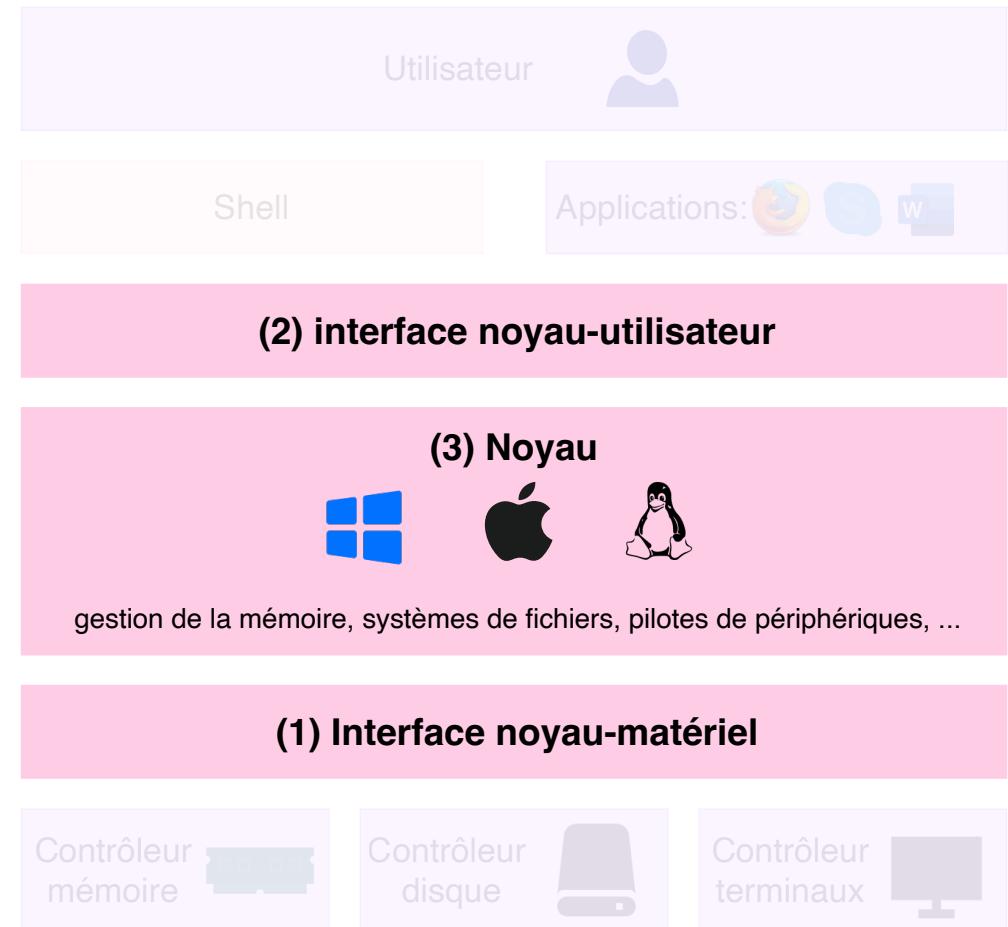
NOYAU D'UN SYSTÈME D'EXPLOITATION

- **Gestion des entrées/sorties (I/O)**
 - contrôleurs, pilotes, ...
- **Gestion des processus**
 - ordonnancement, synchronisation, ...
- **Gestion mémoire**
 - allocation, gestion des espaces, ...



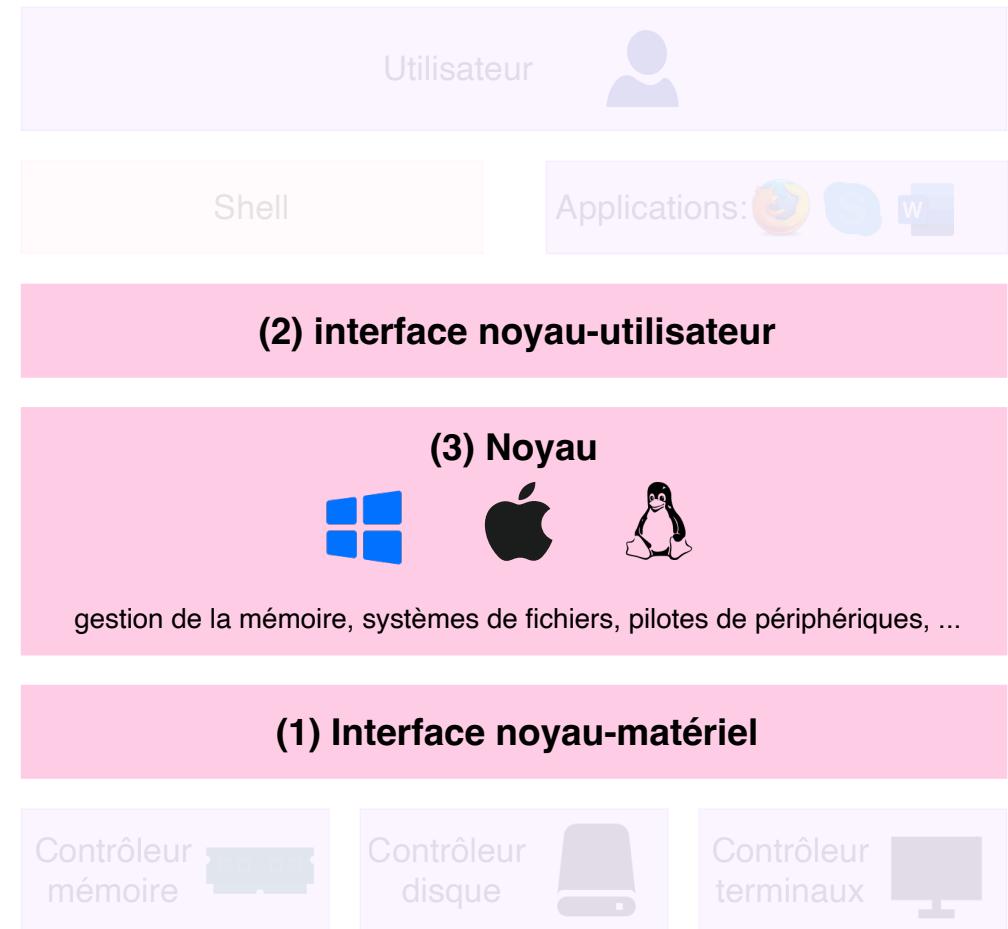
NOYAU D'UN SYSTÈME D'EXPLOITATION

- **Gestion des entrées/sorties (I/O)**
 - contrôleurs, pilotes, ...
- **Gestion des processus**
 - ordonnancement, synchronisation, ...
- **Gestion mémoire**
 - allocation, gestion des espaces, ...
- **Gestion du stockage secondaire**
 - système de fichiers, ...

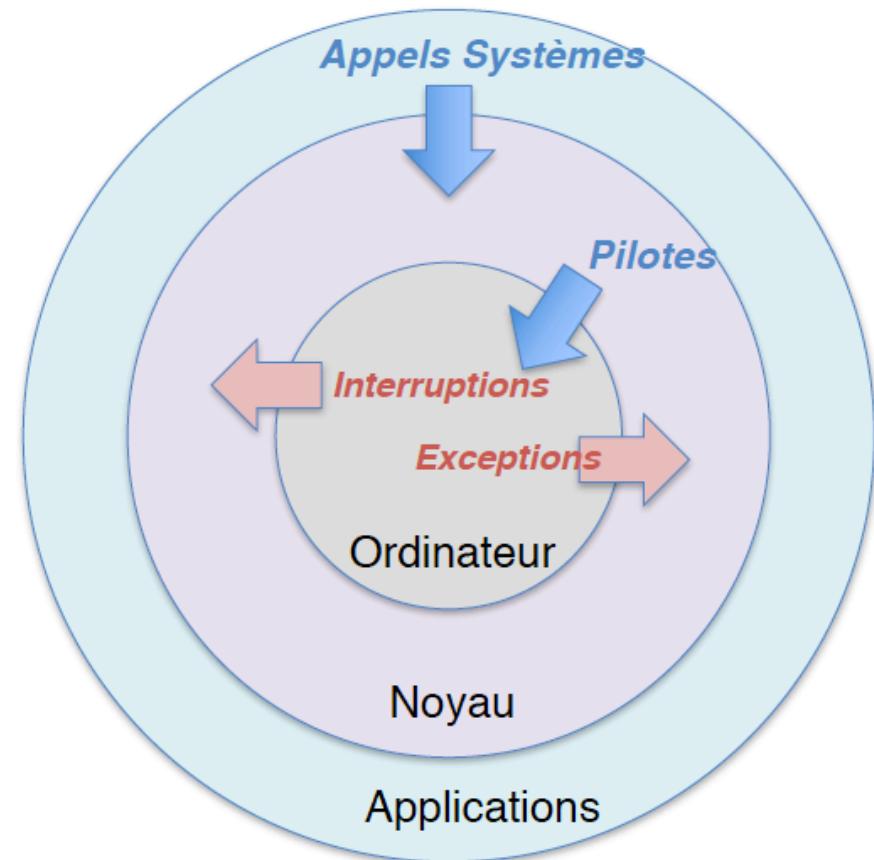


NOYAU D'UN SYSTÈME D'EXPLOITATION

- **Gestion des entrées/sorties (I/O)**
 - contrôleurs, pilotes, ...
- **Gestion des processus**
 - ordonnancement, synchronisation, ...
- **Gestion mémoire**
 - allocation, gestion des espaces, ...
- **Gestion du stockage secondaire**
 - système de fichiers, ...
- **Gestion de la sécurité**

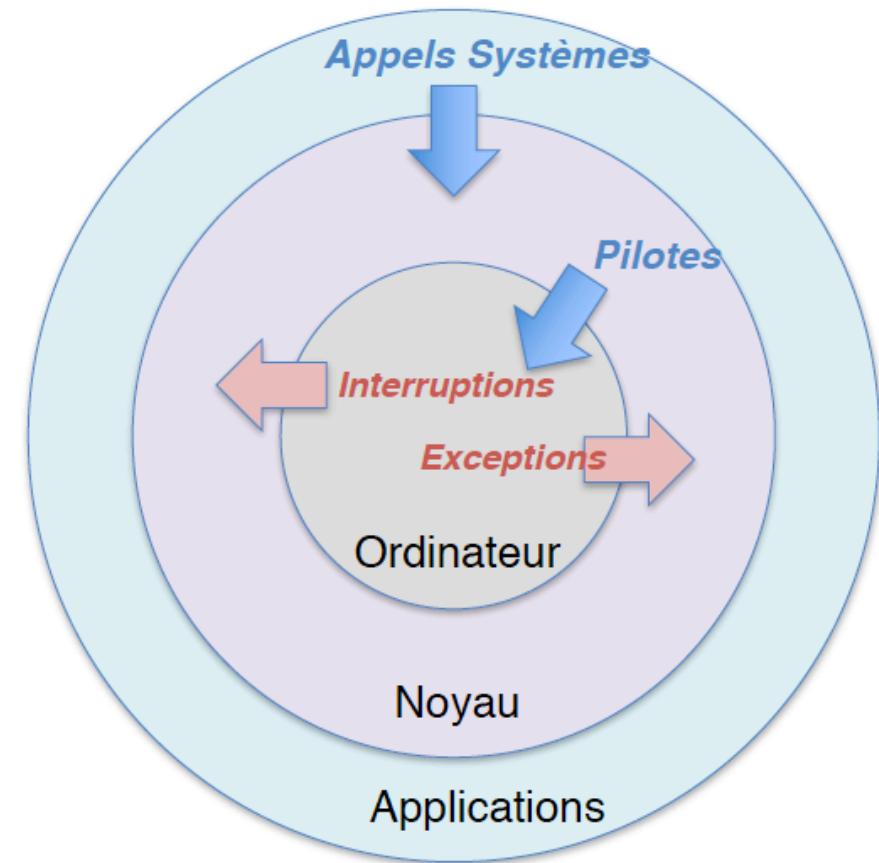


ORGANISATION GÉNÉRALE DE L'OS



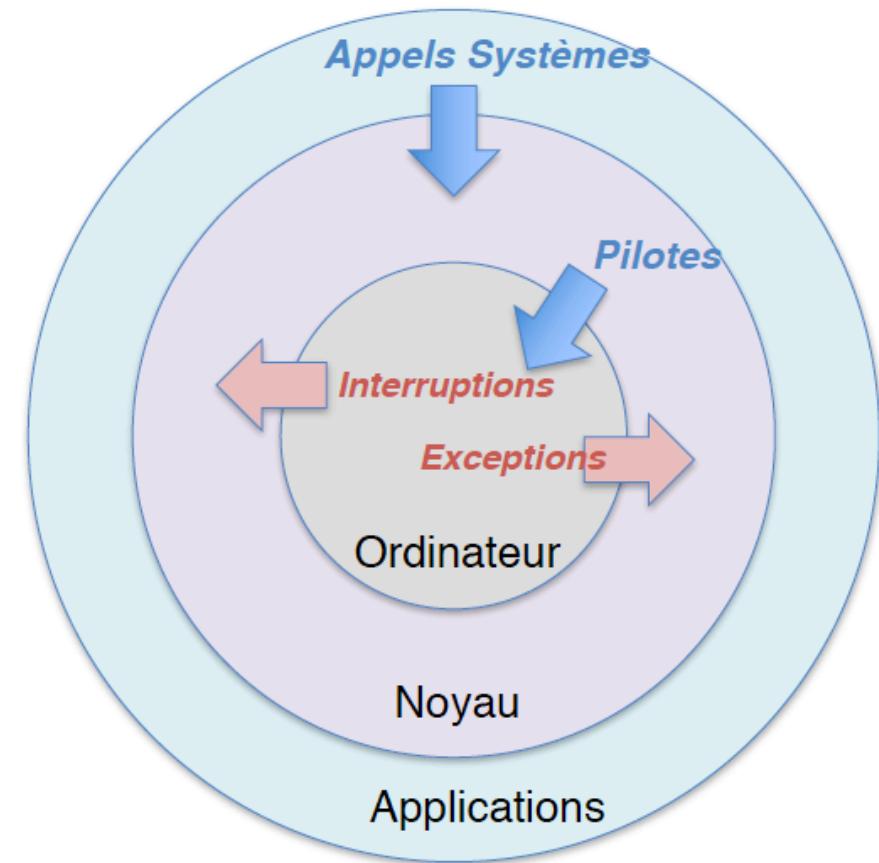
ORGANISATION GÉNÉRALE DE L'OS

- **Interruptions** : évènements produits par le matériel.



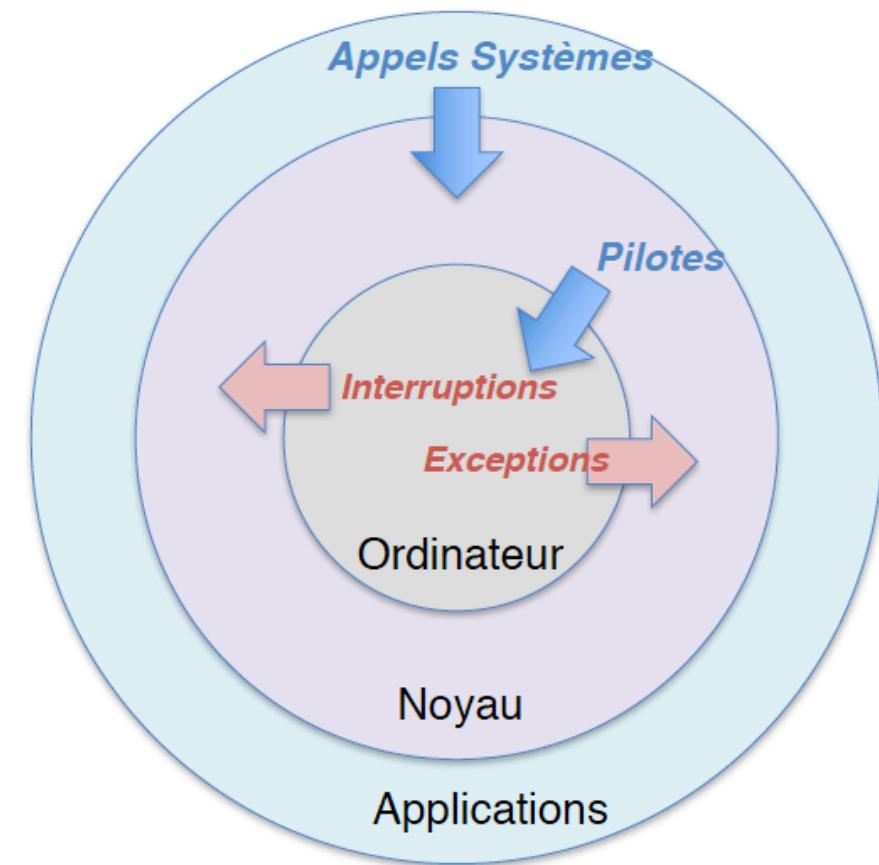
ORGANISATION GÉNÉRALE DE L'OS

- **Interruptions** : évènements produits par le matériel.
- **Exceptions** : événements générés par le processeur.



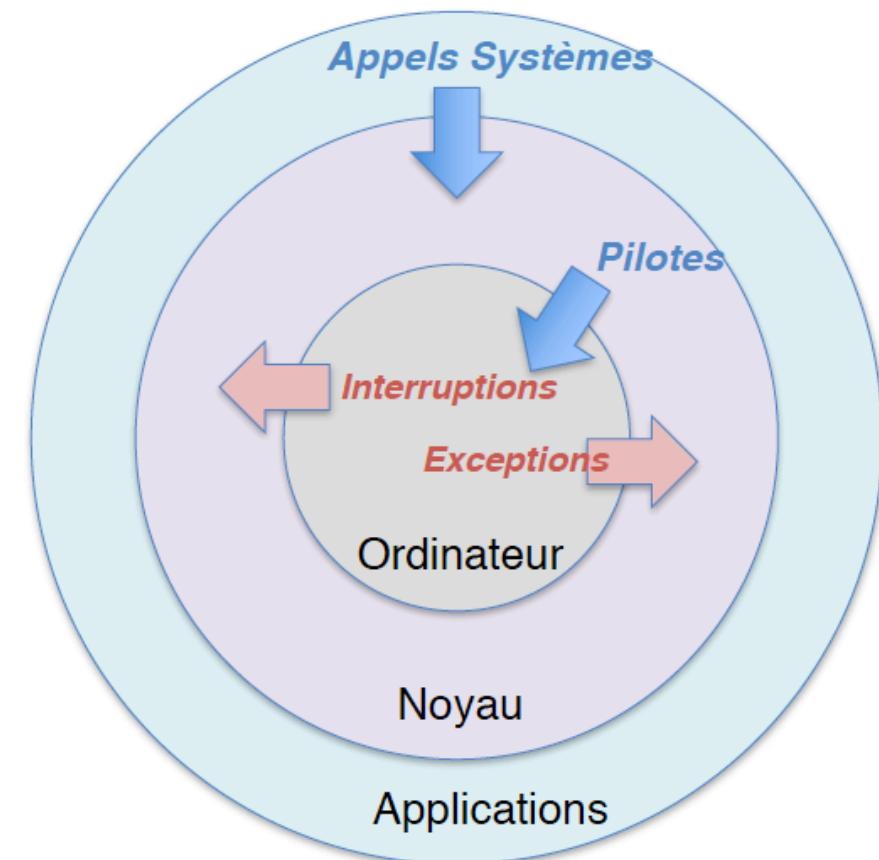
ORGANISATION GÉNÉRALE DE L'OS

- **Interruptions** : évènements produits par le matériel.
- **Exceptions** : événements générés par le processeur.
- **Pilotes (drivers)** : applications contrôlant les périphériques.



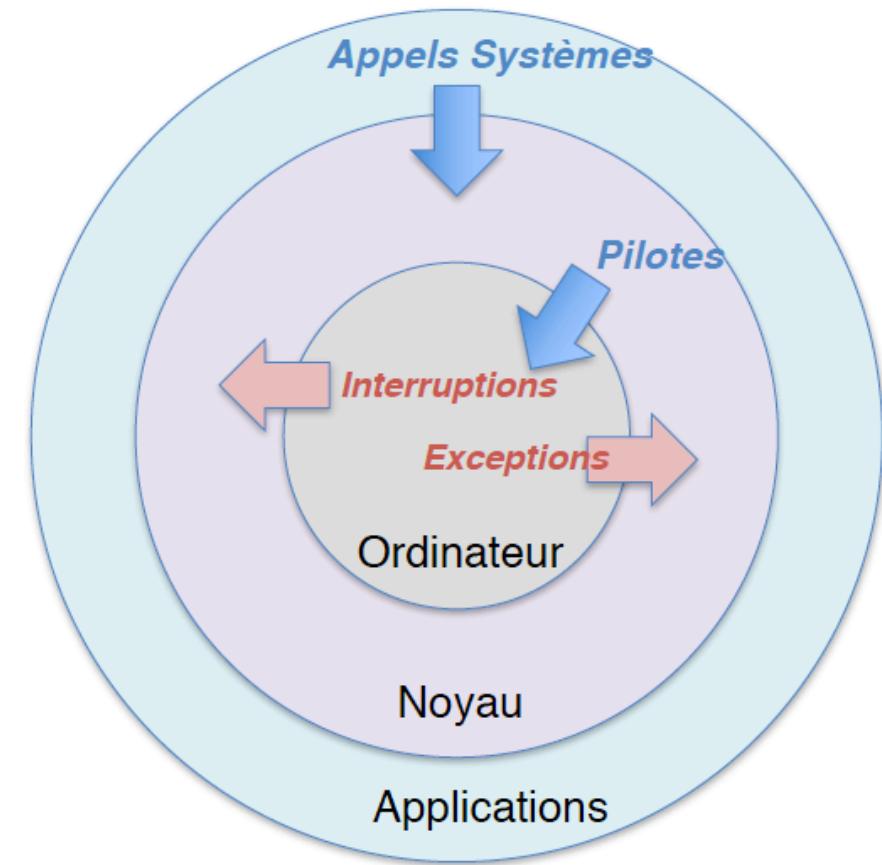
ORGANISATION GÉNÉRALE DE L'OS

- **Interruptions** : évènements produits par le matériel.
- **Exceptions** : événements générés par le processeur.
- **Pilotes (drivers)** : applications contrôlant les périphériques.
- **Noyau (kernel)** : application rendant des services généraux.



ORGANISATION GÉNÉRALE DE L'OS

- **Interruptions** : évènements produits par le matériel.
- **Exceptions** : événements générés par le processeur.
- **Pilotes (drivers)** : applications contrôlant les périphériques.
- **Noyau (kernel)** : application rendant des services généraux.
- **Appels Systèmes** : demandes de services.



STRUCTURE DES OS

STRUCTURE DES OS

- Comment organiser les différentes fonctions d'un OS ?

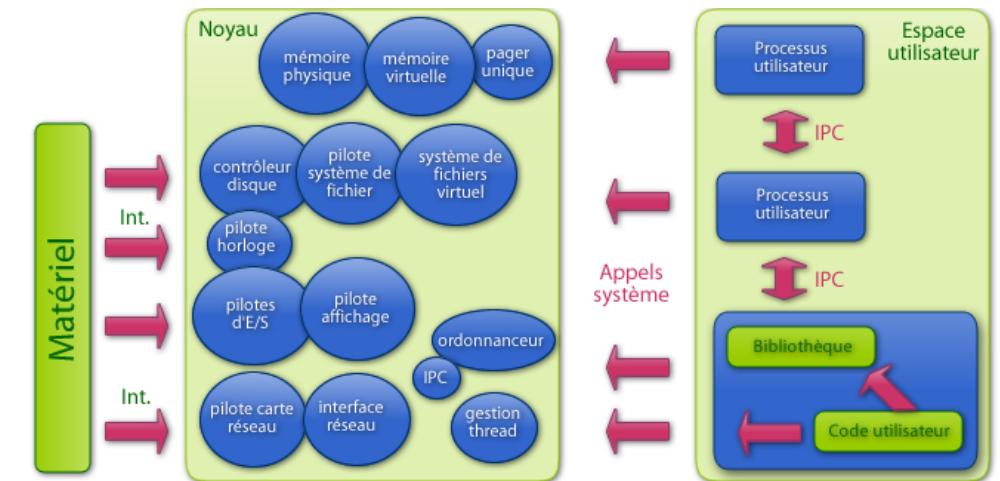
STRUCTURE DES OS

- Comment organiser les différentes fonctions d'un OS ?
👉 Qu'est-ce qui est dans le noyau (en mode Superviseur) ?

STRUCTURE DES OS

- Comment organiser les différentes fonctions d'un OS ?
 - 👉 Qu'est-ce qui est dans le noyau (en mode Superviseur) ?
 - 👉 Comment interagissent les différents composants ?

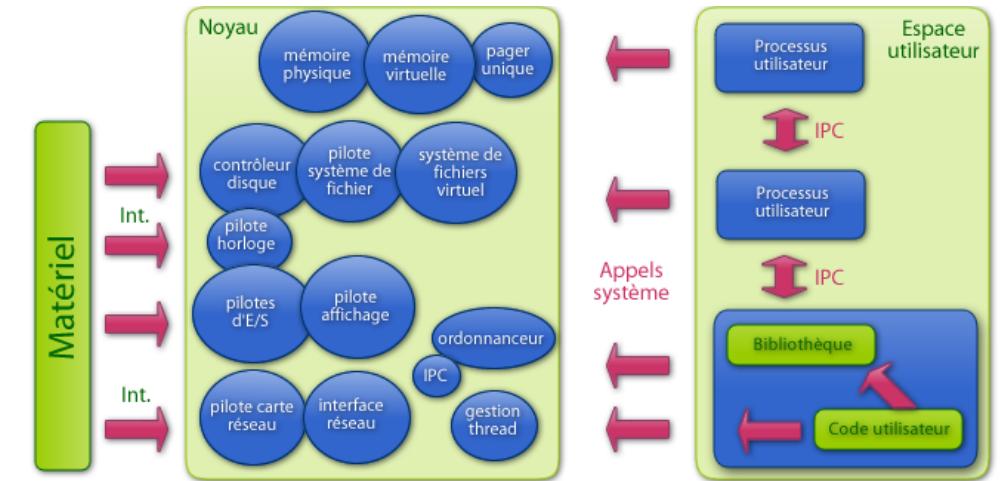
NOYAUX MONOLITHIQUES



source : <https://fr.wikipedia.org>

NOYAUX MONOLITHIQUES

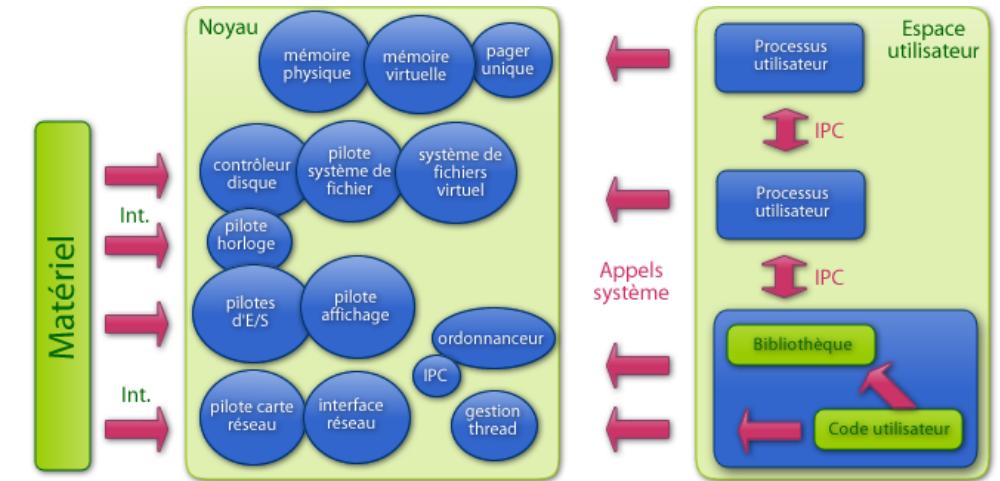
- L'ensemble des fonctions/pilotes sont regroupés dans **un seul bloc**.



source : <https://fr.wikipedia.org>

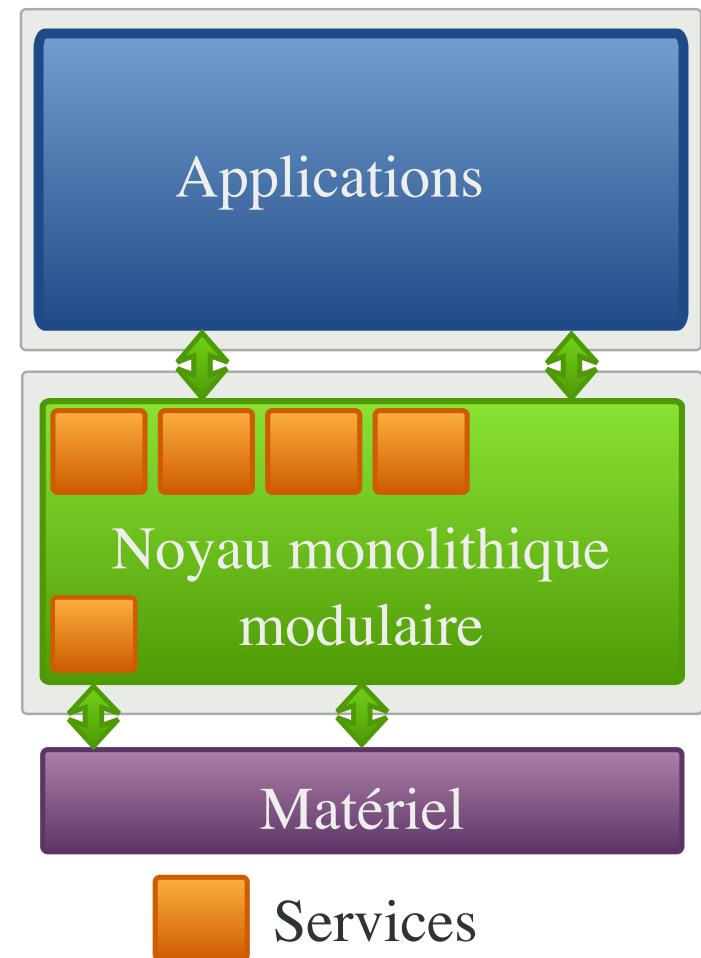
NOYAUX MONOLITHIQUES

- L'ensemble des fonctions/pilotes sont regroupés dans **un seul bloc**.
- Ex. anciennes versions de **Linux** ou certains **vieux Unix**.



source : <https://fr.wikipedia.org>

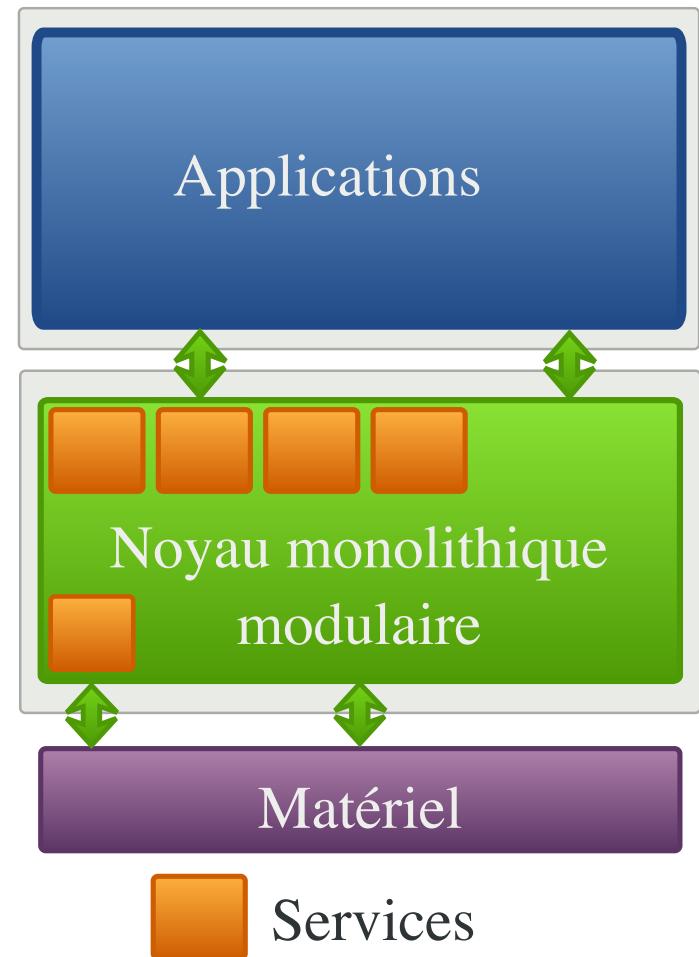
NOYAUX MONOLITHIQUES MODULAIRES



source : <https://fr.wikipedia.org>

NOYAUX MONOLITHIQUES MODULAIRES

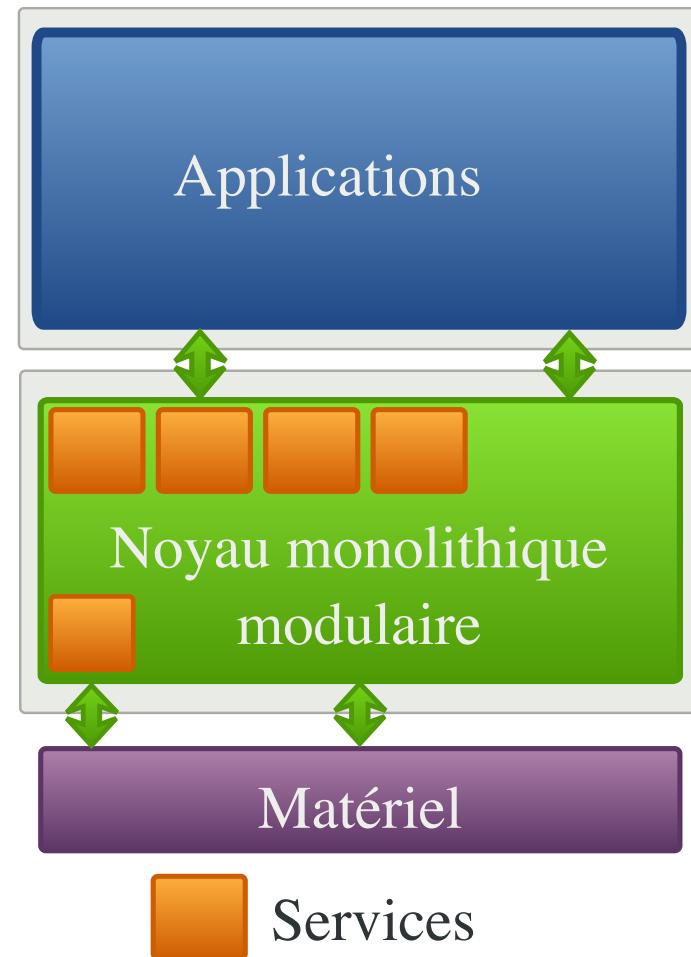
- Seules **les parties fondamentales** de l'OS sont regroupées dans **un bloc unique**.



source : <https://fr.wikipedia.org>

NOYAUX MONOLITHIQUES MODULAIRES

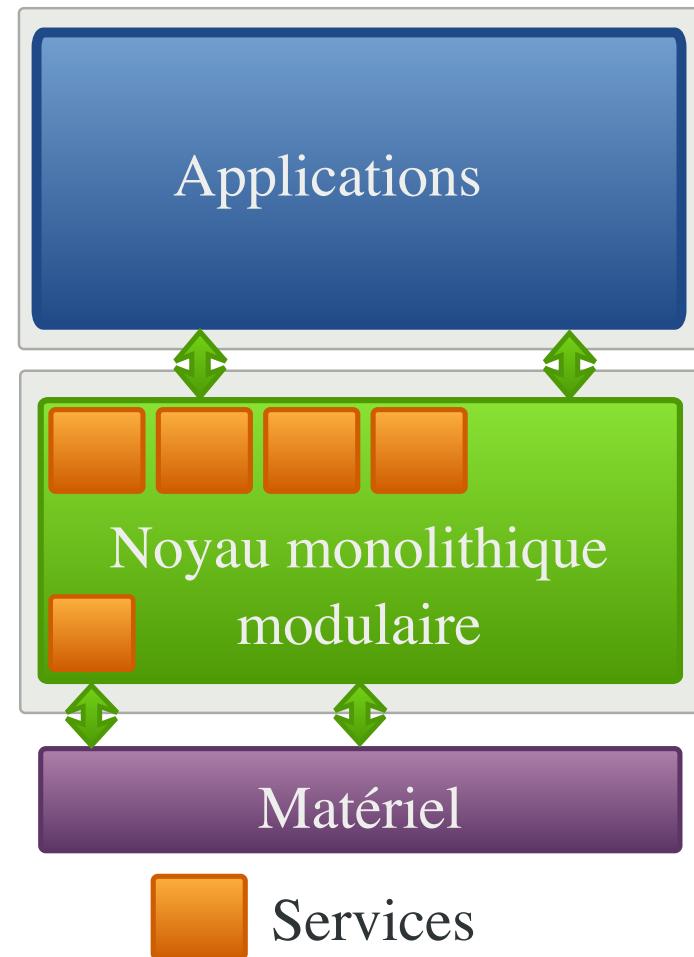
- Seules **les parties fondamentales** de l'OS sont regroupées dans **un bloc unique**.
- Les autres fonctions (**ex.** les pilotes) sont regroupées dans des **modules séparés**.



source : <https://fr.wikipedia.org>

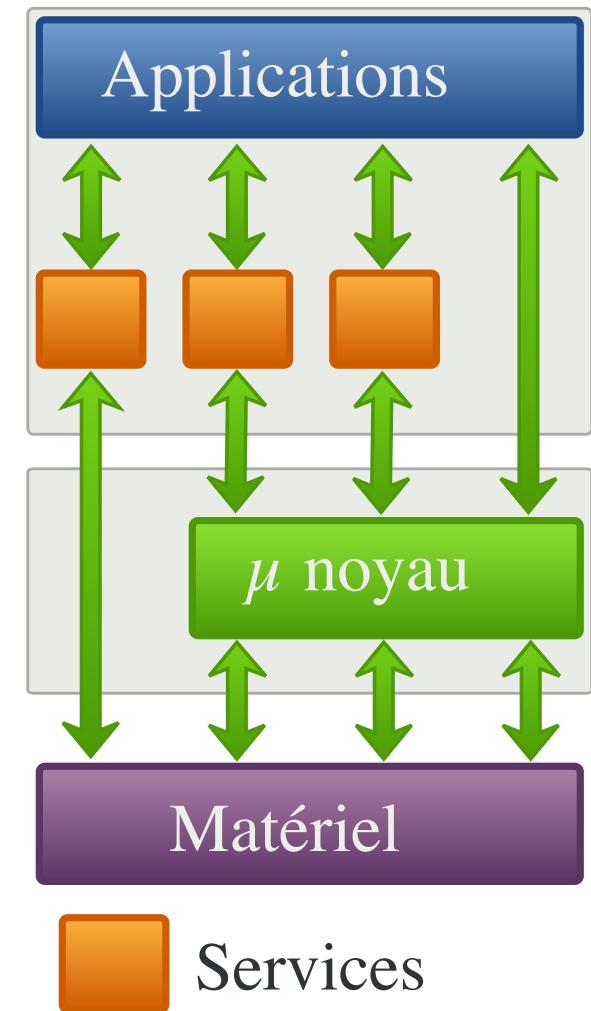
NOYAUX MONOLITHIQUES MODULAIRES

- Seules **les parties fondamentales** de l'OS sont regroupées dans **un bloc unique**.
- Les autres fonctions (**ex.** les pilotes) sont regroupées dans des **modules séparés**.
- **Ex.** Linux ou Solaris.



source : <https://fr.wikipedia.org>

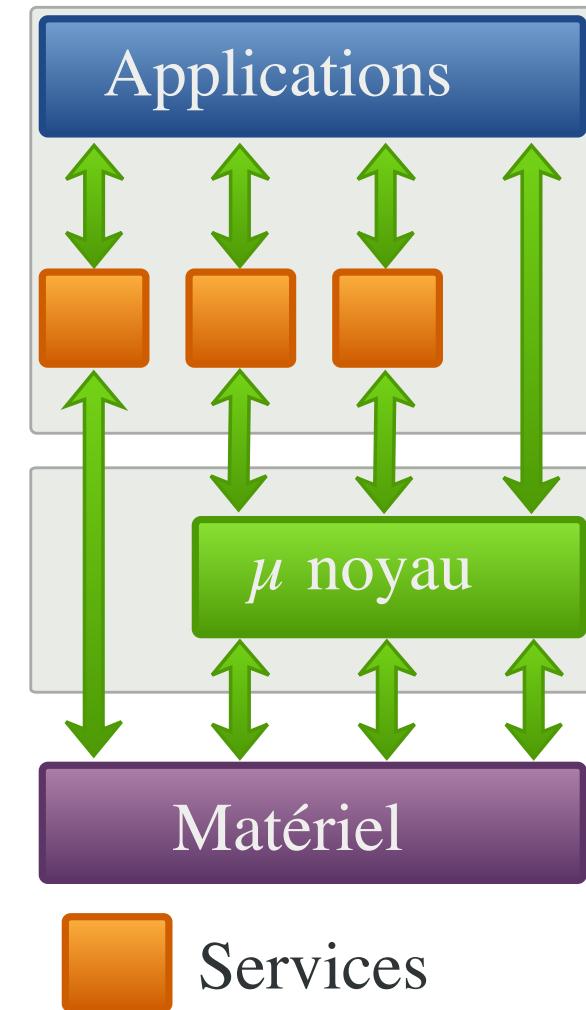
SYSTÈMES À MICRO-NOYAUX



source : <https://fr.wikipedia.org>

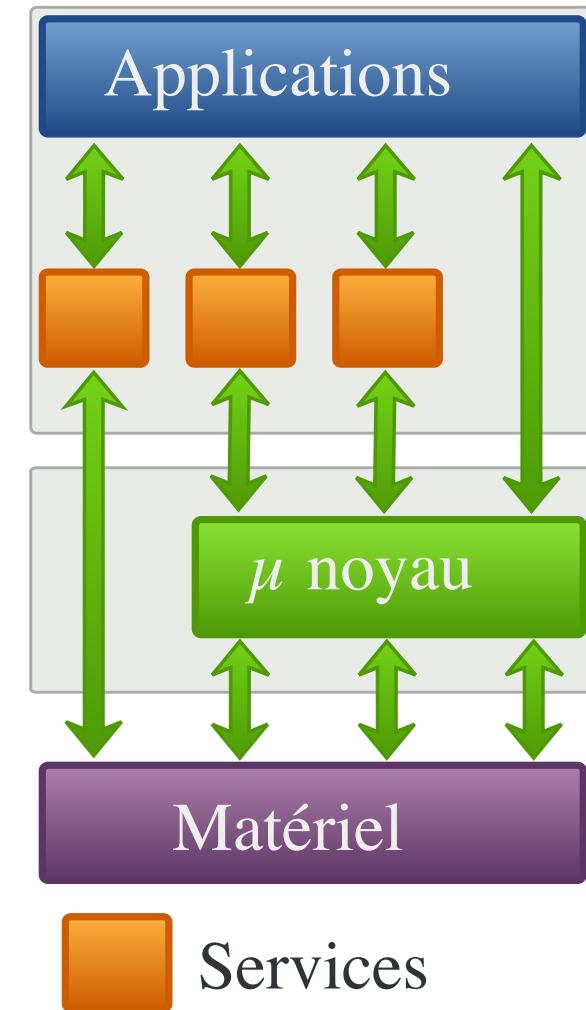
SYSTÈMES À MICRO-NOYAUX

- **Minimiser les fonctionnalités dépendantes** du noyau en plaçant des services l'extérieur.



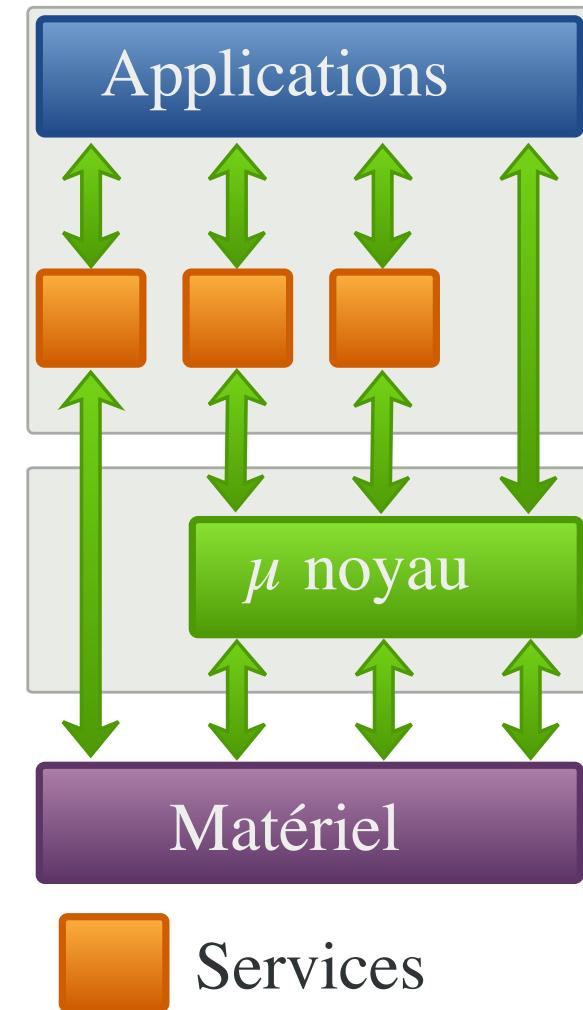
SYSTÈMES À MICRO-NOYAUX

- **Minimiser les fonctionnalités dépendantes** du noyau en plaçant des services l'extérieur.
- **Éloigner les services « à risque »** des parties critiques de l'OS regroupées dans le noyau.

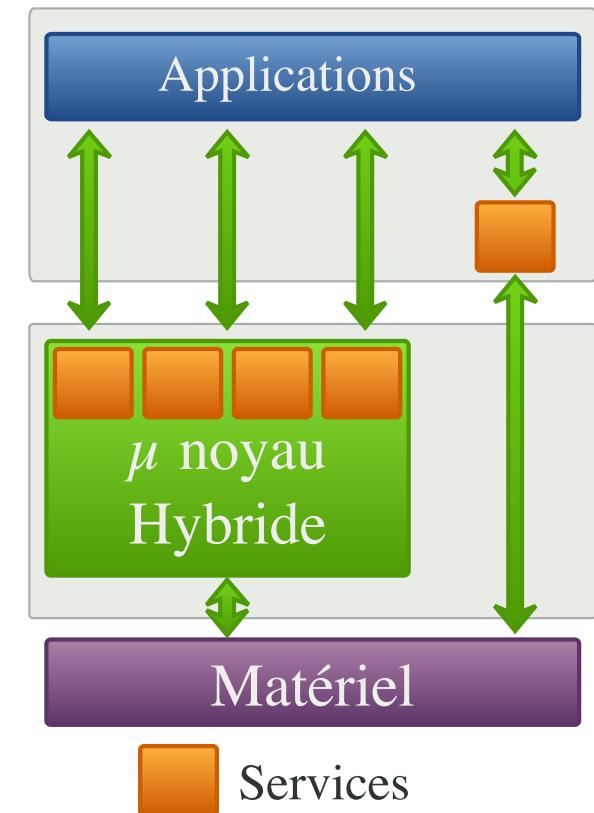


SYSTÈMES À MICRO-NOYAUX

- **Minimiser les fonctionnalités dépendantes** du noyau en plaçant des services l'extérieur.
- **Éloigner les services « à risque »** des parties critiques de l'OS regroupées dans le noyau.
- **Ex. Mach de Mac OS X.**



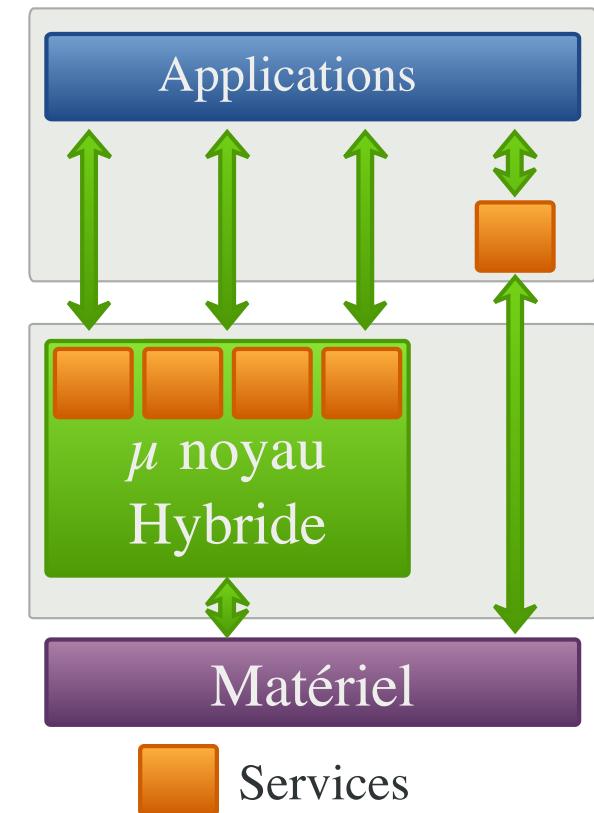
SYSTÈMES À NOYAUX HYBRIDES



source : <https://fr.wikipedia.org>

SYSTÈMES À NOYAUX HYBRIDES

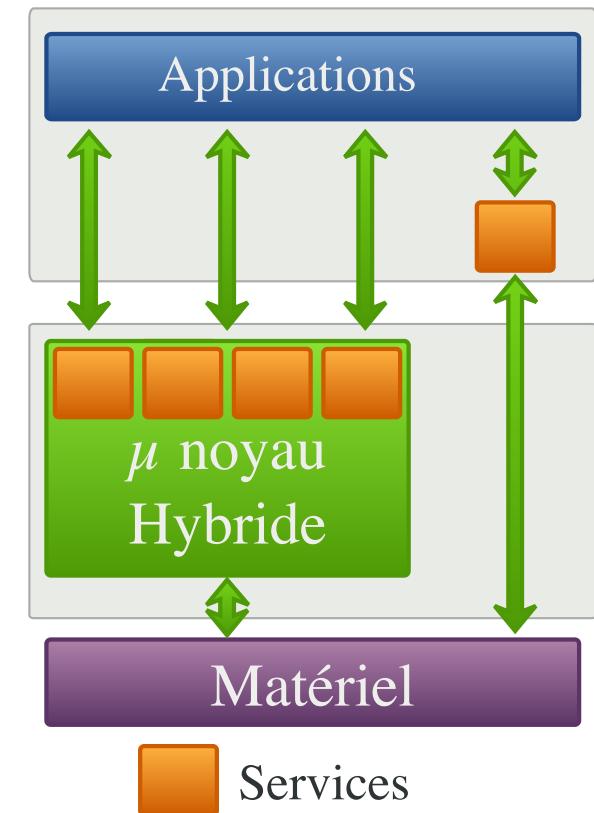
- Reprendre des concepts des noyaux **monolithiques** et des **micro-noyaux** pour combiner **les avantages des deux**.



source : <https://fr.wikipedia.org>

SYSTÈMES À NOYAUX HYBRIDES

- Reprendre des concepts des noyaux **monolithiques** et des **micro-noyaux** pour combiner **les avantages des deux**.
- Ex. XNU de **Mac OS X**.



source : <https://fr.wikipedia.org>

CONCLUSIONS

CONCLUSIONS

- Les OS **monolithiques** sont rapides mais délicats à maintenir.

CONCLUSIONS

- Les OS **monolithiques** sont rapides mais délicats à maintenir.
- Les OS **monolithiques modulaires** ne sont pas faciles à concevoir (dépendances multiples).

CONCLUSIONS

- Les OS **monolithiques** sont rapides mais délicats à maintenir.
- Les OS **monolithiques modulaires** ne sont pas faciles à concevoir (dépendances multiples).
- Les OS à **micro-noyaux** pur sont trop lents.

CONCLUSIONS

- Les OS **monolithiques** sont rapides mais délicats à maintenir.
- Les OS **monolithiques modulaires** ne sont pas faciles à concevoir (dépendances multiples).
- Les OS à **micro-noyaux** pur sont trop lents.

- **Les tendances :**

CONCLUSIONS

- Les OS **monolithiques** sont rapides mais délicats à maintenir.
- Les OS **monolithiques modulaires** ne sont pas faciles à concevoir (dépendances multiples).
- Les OS à **micro-noyaux** pur sont trop lents.

- **Les tendances :**
 - 👉 OS à **noyaux hybrides**.

CONCLUSIONS

- Les OS **monolithiques** sont rapides mais délicats à maintenir.
- Les OS **monolithiques modulaires** ne sont pas faciles à concevoir (dépendances multiples).
- Les OS à **micro-noyaux** pur sont trop lents.

- **Les tendances :**
 - 👉 OS à **noyaux hybrides**.
 - 👉 un micro-noyau étendus en fonctionnalités par d'autres composants.

CHARGEMENT D'UN OS

CHARGEMENT D'UN OS

- L'**OS** est le **premier programme exécuté** lors de la mise en marche de l'ordinateur, après l'amorçage (**boot**).

CHARGEMENT D'UN OS

- L'**OS** est le **premier programme exécuté** lors de la mise en marche de l'ordinateur, après l'amorçage (**boot**).
- Le **boot (bootstrap)** désigne les **étapes successives du démarrage**.

LES ÉTAPES DU BOOT

LES ÉTAPES DU BOOT

1. le **POST** test - Power On Self Test

LES ÉTAPES DU BOOT

1. le POST test - Power On Self Test

- après un start ou un reset, le processeur charge **les premières instructions** à partir de la **ROM** du **BIOS** situées à l'adresse **FFFF0**.

LES ÉTAPES DU BOOT

1. le POST test - Power On Self Test

- après un start ou un reset, le processeur charge **les premières instructions** à partir de la **ROM** du **BIOS** situées à l'adresse **FFFF0**.
- des instructions de branchement vers un programme du **BIOS** qui initialise et teste les fonctions vitales du hardware

LES ÉTAPES DU BOOT

1. le POST test - Power On Self Test

- après un start ou un reset, le processeur charge les premières instructions à partir de la ROM du BIOS situées à l'adresse FFFF0.
- des instructions de branchement vers un programme du BIOS qui initialise et teste les fonctions vitales du hardware

2. le chargement du MBR - Master Boot Record

LES ÉTAPES DU BOOT

1. le POST test - Power On Self Test

- après un start ou un reset, le processeur charge les premières instructions à partir de la ROM du BIOS situées à l'adresse FFFF0.
- des instructions de branchement vers un programme du BIOS qui initialise et teste les fonctions vitales du hardware

2. le chargement du MBR - Master Boot Record

- si le POST réussit, il consultera la RAM CMOS pour identifier le disque système dont le premier secteur est appelé MBR.

LES ÉTAPES DU BOOT

1. le POST test - Power On Self Test

- après un start ou un reset, le processeur charge les premières instructions à partir de la ROM du BIOS situées à l'adresse FFFF0.
- des instructions de branchement vers un programme du BIOS qui initialise et teste les fonctions vitales du hardware

2. le chargement du MBR - Master Boot Record

- si le POST réussit, il consultera la RAM CMOS pour identifier le disque système dont le premier secteur est appelé MBR.
- le code du MBR teste la table de partition pour charger la partition contenant le secteur d'amorçage avec l'IPL - Initial Program Load.

LES ÉTAPES DU BOOT

1. le POST test - Power On Self Test

- après un start ou un reset, le processeur charge les premières instructions à partir de la ROM du BIOS situées à l'adresse FFFF0.
- des instructions de branchement vers un programme du BIOS qui initialise et teste les fonctions vitales du hardware

2. le chargement du MBR - Master Boot Record

- si le POST réussit, il consultera la RAM CMOS pour identifier le disque système dont le premier secteur est appelé MBR.
- le code du MBR teste la table de partition pour charger la partition contenant le secteur d'amorçage avec l'IPL - Initial Program Load.
 - 👉 l'IPL charge l'OS ou le bootmanager en RAM.
 - 👉 l'OS est lancé

PLAN

- Architecture des ordinateurs
- Structure et évolution des OS
- Fonctionnalités principales d'un OS
- Virtualisation

[Retour au plan](#) - [Retour à l'accueil](#)

FONCTIONNALITÉS PRINCIPALES D'UN OS

FONCTIONNALITÉS PRINCIPALES D'UN OS

1. Gestion des processus

FONCTIONNALITÉS PRINCIPALES D'UN OS

1. Gestion des processus
2. Gestion de la mémoire

FONCTIONNALITÉS PRINCIPALES D'UN OS

1. Gestion des processus
2. Gestion de la mémoire
3. Mémoire virtuelle

FONCTIONNALITÉS PRINCIPALES D'UN OS

1. Gestion des processus
2. Gestion de la mémoire
3. Mémoire virtuelle
4. Système de fichiers

FONCTIONNALITÉS PRINCIPALES D'UN OS

1. Gestion des processus
2. Gestion de la mémoire
3. Mémoire virtuelle
4. Système de fichiers
5. Sécurité

GESTION DES PROCESSUS

⌚ Fonctionnalités principales d'un OS

DÉFINITIONS

DÉFINITIONS

- Un **programme informatique** : une suite statique d'instructions.

DÉFINITIONS

- Un **programme informatique** : une suite statique d'instructions.
- Un **processeur** : un **automate** (électronique) de **traitement**.
 - Il peut **exécuter** un programme
 - Il **modifie son état** en fonction des instructions

DÉFINITIONS

- Un **programme informatique** : une suite statique d'instructions.
- Un **processeur** : un **automate** (électronique) de **traitement**.
 - Il peut **exécuter** un programme
 - Il **modifie son état** en fonction des instructions
- Un **processus** : un programme exécuté par un processeur.

DÉFINITIONS

- Un **programme informatique** : une suite statique d'instructions.
- Un **processeur** : un **automate** (électronique) de **traitement**.
 - Il peut **exécuter** un programme
 - Il **modifie son état** en fonction des instructions
- Un **processus** : un programme exécuté par un processeur.
 - capte le **caractère dynamique** d'un programme.

PROGRAMME vs PROCESSUS

- Un **processus** : un programme exécuté par un processeur.
 - capte le **caractère dynamique** d'un programme.

PROGRAMME vs PROCESSUS

- Un **processus** : un programme exécuté par un processeur.
 - capte le **caractère dynamique** d'un programme.
- **Un programme** peut donner **naissance à plusieurs processus**.

PROGRAMME vs PROCESSUS

- Un **processus** : un programme exécuté par un processeur.
 - capte le **caractère dynamique** d'un programme.
- **Un programme** peut donner **naissance à plusieurs processus**.
- **Un processus** est forcément **créé par un autre processus** (le système d'exploitation par exemple)

PROGRAMME vs PROCESSUS

- Un **processus** : un programme exécuté par un processeur.
 - capte le **caractère dynamique** d'un programme.
- **Un programme** peut donner **naissance à plusieurs processus**.
- **Un processus** est forcément **créé par un autre processus** (le système d'exploitation par exemple)

Exemples de processus

- Logiciel de traitement de texte
- Compilation de code source
- Tâche système (envoi de données vers l'imprimante)

PROCESSUS

PROCESSUS

- Dans **un OS moderne**, plusieurs processus **s'exécutent en parallèle** :

PROCESSUS

- Dans **un OS moderne**, plusieurs processus **s'exécutent en parallèle** :
 - **Les processus de l'OS** (gestion du réseau, gestion des utilisateurs, ...)

PROCESSUS

- Dans **un OS moderne**, plusieurs processus **s'exécutent en parallèle** :
 - **Les processus de l'OS** (gestion du réseau, gestion des utilisateurs, ...)
 - Le **shell** (toute l'**interface graphique** → plusieurs processus).

PROCESSUS

- Dans **un OS moderne**, plusieurs processus **s'exécutent en parallèle** :
 - **Les processus de l'OS** (gestion du réseau, gestion des utilisateurs, ...)
 - Le **shell** (toute **l'interface graphique** → plusieurs processus).
 - L'**IDE VSCode** avec lequel je tape ce cours.

PROCESSUS

- Dans **un OS moderne**, plusieurs processus **s'exécutent en parallèle** :
 - **Les processus de l'OS** (gestion du réseau, gestion des utilisateurs, ...)
 - Le **shell** (toute **l'interface graphique** → plusieurs processus).
 - L'**IDE VSCode** avec lequel je tape ce cours.
 - Le navigateur **Chrome** qui me permet de visualiser ce cours.

PROCESSUS

- Dans **un OS moderne**, plusieurs processus **s'exécutent en parallèle** :
 - **Les processus de l'OS** (gestion du réseau, gestion des utilisateurs, ...)
 - Le **shell** (toute **l'interface graphique** → plusieurs processus).
 - L'**IDE VSCode** avec lequel je tape ce cours.
 - Le navigateur **Chrome** qui me permet de visualiser ce cours.

```
$ top -stats command,pid,ppid,cpu,pstate

COMMAND      PID  PPID  %CPU  STATE
com.docker.hyper    674    667   34.9  sleeping
launchd          1      0    8.5  sleeping
top            74562  34386   3.7  running
Terminal        34311      1    3.0  sleeping
WindowServer     169      1    2.5  sleeping
...
```

RÔLE DE L'OS

RÔLE DE L'OS

- **Création et suppression de processus**
 - Programme → processus
 - Munir le programme des **informations** nécessaires pour son **exécution**

RÔLE DE L'OS

- **Création et suppression de processus**
 - Programme → processus
 - **Munir** le programme des **informations** nécessaires pour son **exécution**
- **Suspension et reprise**
 - Multiprogrammation → **interrompre et reprendre** les processus
 - **Gestion de la mémoire** où sont stockées les processus interrompus

RÔLE DE L'OS

- **Création et suppression de processus**
 - Programme → processus
 - **Munir** le programme des **informations** nécessaires pour son **exécution**
- **Suspension et reprise**
 - Multiprogrammation → **interrompre et reprendre** les processus
 - **Gestion de la mémoire** où sont stockées les processus interrompus
- **Communication et synchronisation**
 - Partage de données entre plusieurs processus
 - **Consistance** de l'état de la mémoire

RÔLE DE L'OS

- **Création et suppression de processus**
 - Programme → processus
 - **Munir** le programme des **informations** nécessaires pour son **exécution**
- **Suspension et reprise**
 - Multiprogrammation → **interrompre et reprendre** les processus
 - **Gestion de la mémoire** où sont stockées les processus interrompus
- **Communication et synchronisation**
 - Partage de données entre plusieurs processus
 - **Consistance** de l'état de la mémoire

RÔLE DE L'OS

- **Création et suppression de processus**
 - Programme → processus
 - Munir le programme des **informations** nécessaires pour son **exécution**
- **Suspension et reprise**
 - Multiprogrammation → **interrompre et reprendre** les processus
 - **Gestion de la mémoire** où sont stockées les processus interrompus
- **Communication et synchronisation**
 - Partage de données entre plusieurs processus
 - **Consistance** de l'état de la mémoire

CRÉATION DE PROCESSUS

CRÉATION DE PROCESSUS

Rappel : un processus est forcément créé par un autre processus

CRÉATION DE PROCESSUS

Rappel : un processus est forcément créé par un autre processus

- **Sous UNIX** → 2 appels système

CRÉATION DE PROCESSUS

Rappel : un processus est forcément créé par un autre processus

- **Sous UNIX** → 2 appels système
 1. **fork** pour créer un processus à partir du processus courant👉 le processus courant est **duplicé**

CRÉATION DE PROCESSUS

Rappel : un processus est forcément créé par un autre processus

- **Sous UNIX** → 2 appels système
 1. **fork** pour créer un processus à partir du processus courant
👉 le processus courant est **duplicé**
 2. **exec** pour **remplacer** le processus courant par un autre processus

CRÉATION DE PROCESSUS

Rappel : un processus est forcément créé par un autre processus

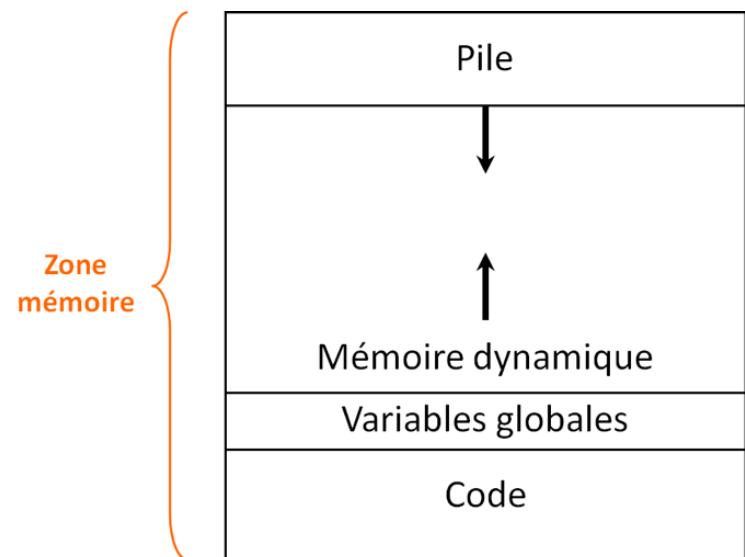
- **Sous UNIX** → 2 appels système
 1. `fork` pour créer un processus à partir du processus courant
👉 le processus courant est **duplicé**
 2. `exec` pour **remplacer** le processus courant par un autre processus
- **Sous WINDOWS**

CRÉATION DE PROCESSUS

Rappel : un processus est forcément créé par un autre processus

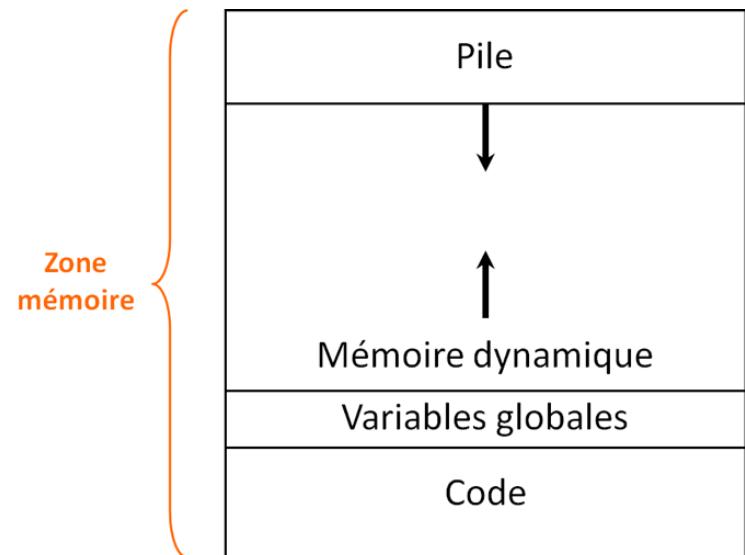
- **Sous UNIX** → 2 appels système
 1. `fork` pour créer un processus à partir du processus courant
👉 le processus courant est **duplicé**
 2. `exec` pour **remplacer** le processus courant par un autre processus
- **Sous WINDOWS**
 - `createprocess` pour créer un processus (**cf. exec Unix**)
👉 le processus courant est **conservé**

L'ESPACE MÉMOIRE D'UN PROCESSUS



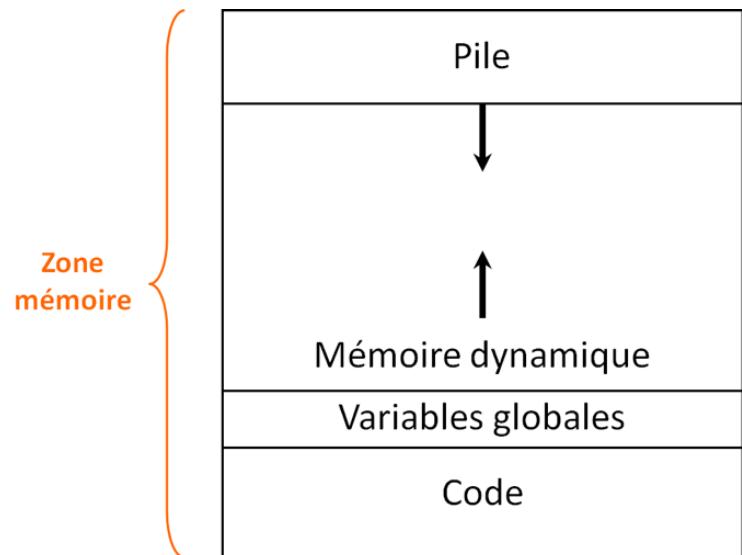
L'ESPACE MÉMOIRE D'UN PROCESSUS

- **Code exécutable** en lecture seule
(taille connue)



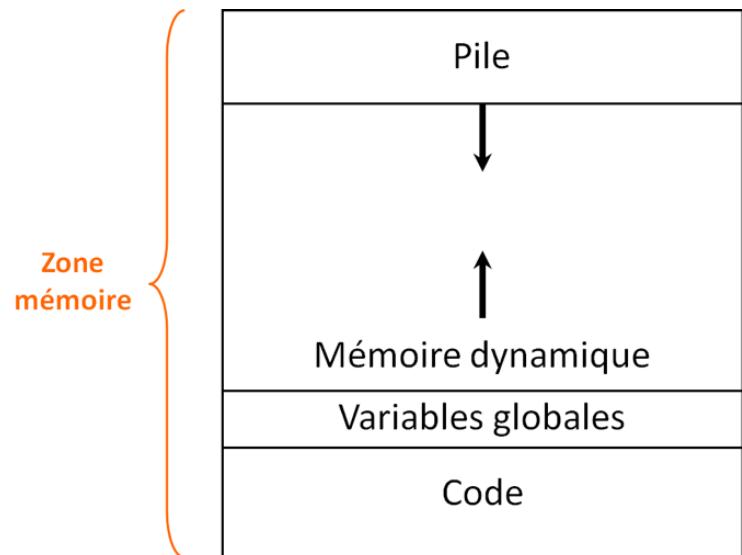
L'ESPACE MÉMOIRE D'UN PROCESSUS

- **Code exécutable** en lecture seule
(taille connue)
- **Variables/Constantes globales**
(taille connue)

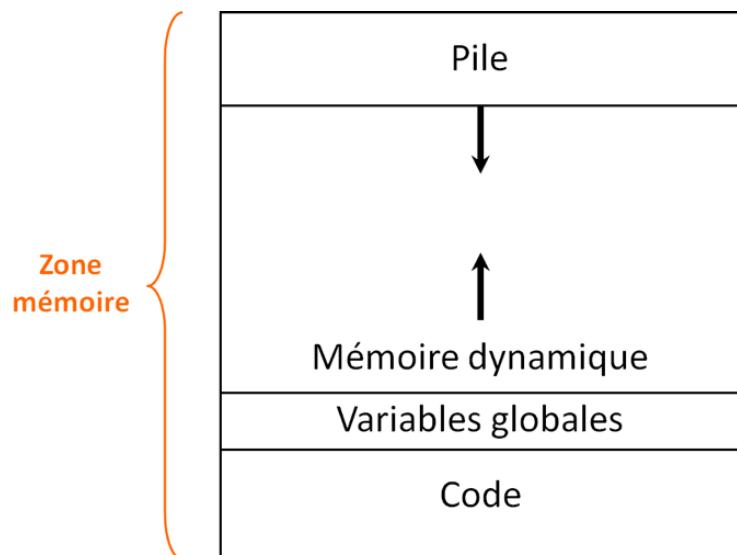


L'ESPACE MÉMOIRE D'UN PROCESSUS

- **Code exécutable** en lecture seule (taille connue)
- **Variables/Constantes globales** (taille connue)
- **Pile** pour gérer les contextes et les variables temporaires (taille inconnue)



L'ESPACE MÉMOIRE D'UN PROCESSUS



- **Code exécutable** en lecture seule (taille connue)
- **Variables/Constantes globales** (taille connue)
- **Pile** pour gérer les contextes et les variables temporaires (taille inconnue)
- **Le TAS ou la Zone d'allocation dynamique de mémoire** (taille inconnue)

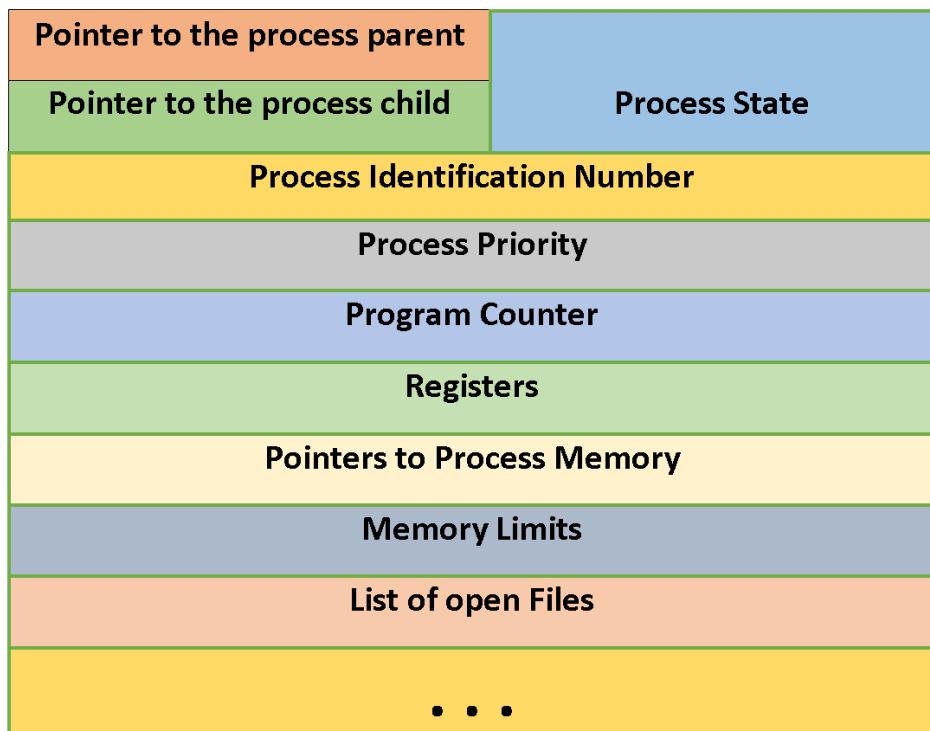
BLOC DE CONTRÔLE DE PROCESSUS

BLOC DE CONTRÔLE DE PROCESSUS

- Process Control Block - PCB
 - Structure de données contenant les informations relatives à un processus utilisée par l'OS pour la gestion des processus.

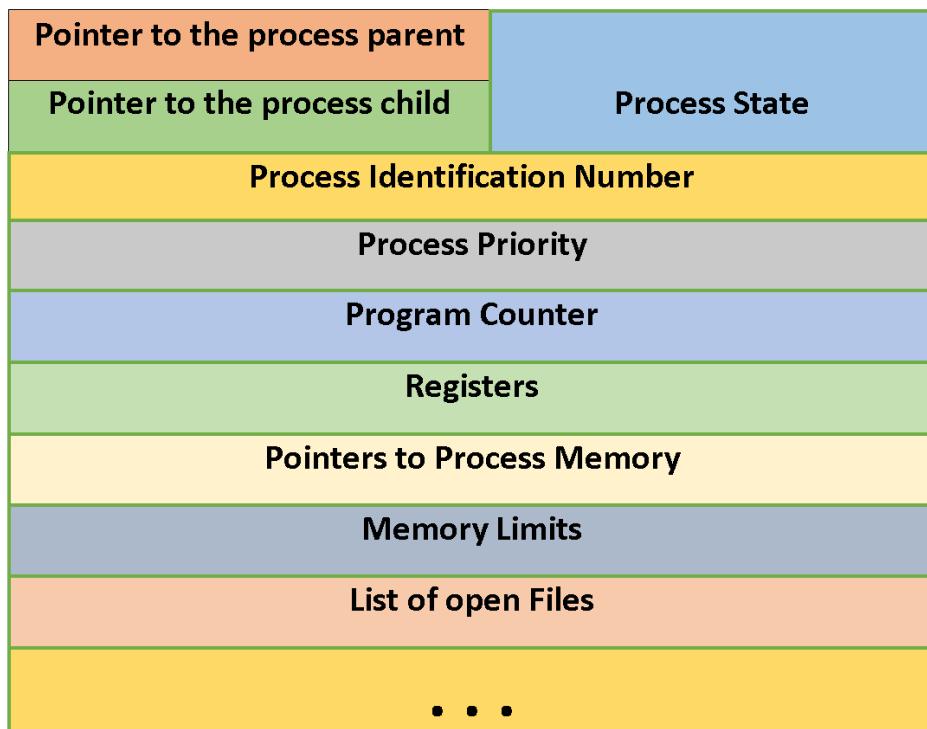
BLOC DE CONTRÔLE DE PROCESSUS

- **Process Control Block - PCB**
 - **Structure de données** contenant les informations relatives à un processus utilisée par l'OS pour la **gestion des processus**.



BLOC DE CONTRÔLE DE PROCESSUS

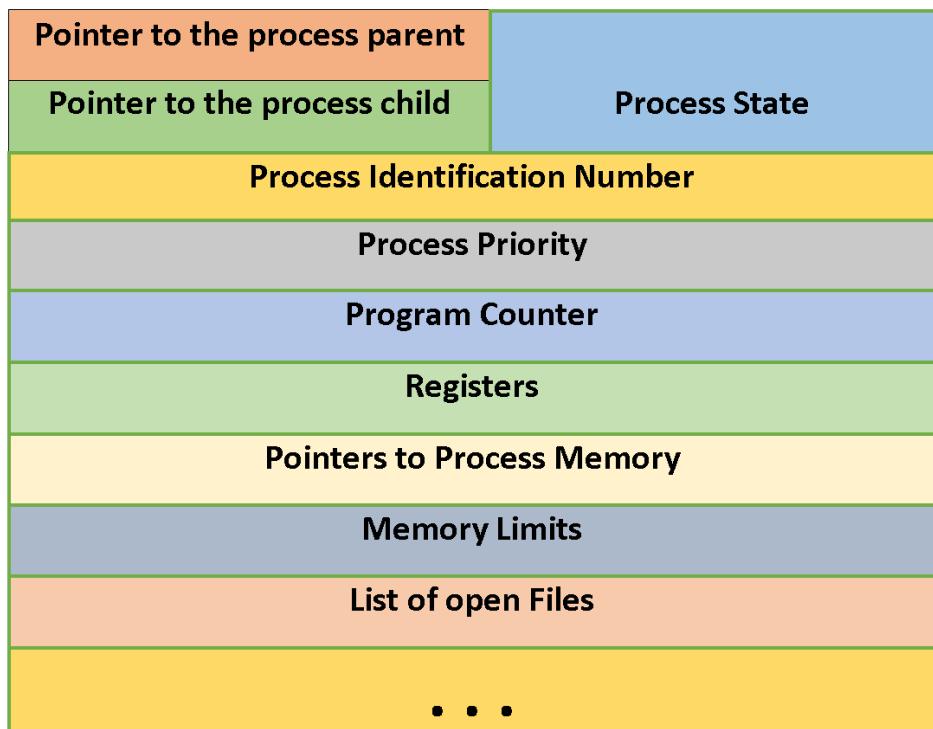
- **Process Control Block - PCB**
 - **Structure de données** contenant les informations relatives à un processus utilisée par l'OS pour la **gestion des processus**.



- Information **mémoire**
- Données d'**ordonnancement**
- **Périphériques** alloués
- Information d'**usage**
- **État** du processus/processeur

BLOC DE CONTRÔLE DE PROCESSUS

- **Process Control Block - PCB**
 - **Structure de données** contenant les informations relatives à un processus utilisée par l'OS pour la **gestion des processus**.



Tout ce qui doit être sauvegardé pour **interrompre** puis **reprendre** l'exécution d'un processus.

RÔLE DE L'OS

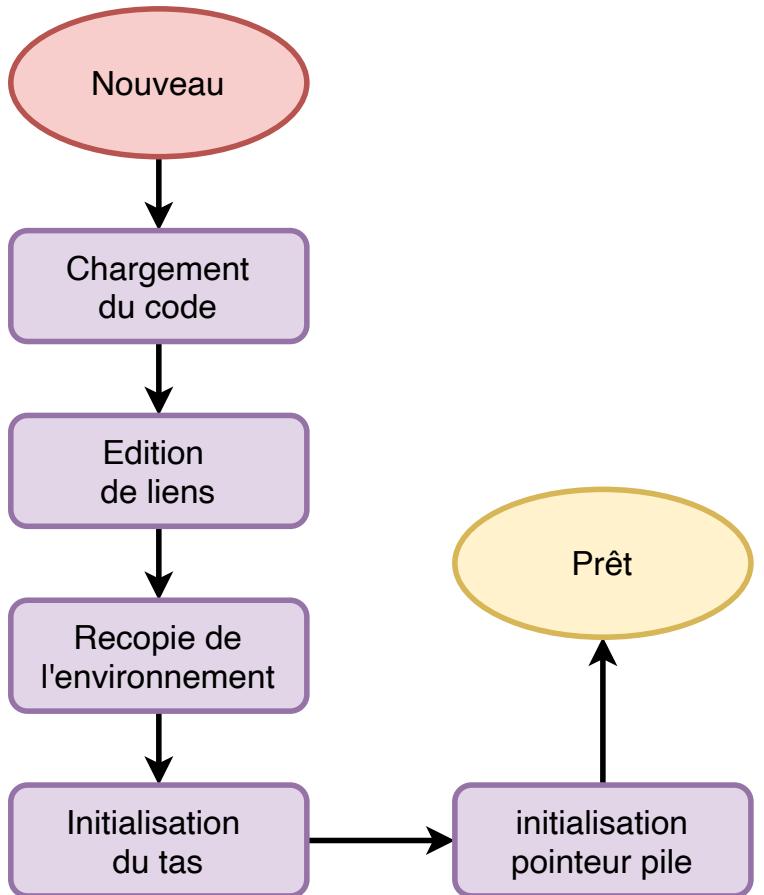
- **Création et suppression de processus**
 - Programme → processus
 - **Munir** le programme des **informations** nécessaires pour son **exécution**
- **Suspension et reprise**
 - Multiprogrammation → **interrompre et reprendre** les processus
 - **Gestion de la mémoire** où sont stockées les processus interrompus
- **Communication et synchronisation**
 - Partage de données entre plusieurs processus
 - **Consistance** de l'état de la mémoire

CYCLE DE VIE DU PROCESSUS

CYCLE DE VIE DU PROCESSUS

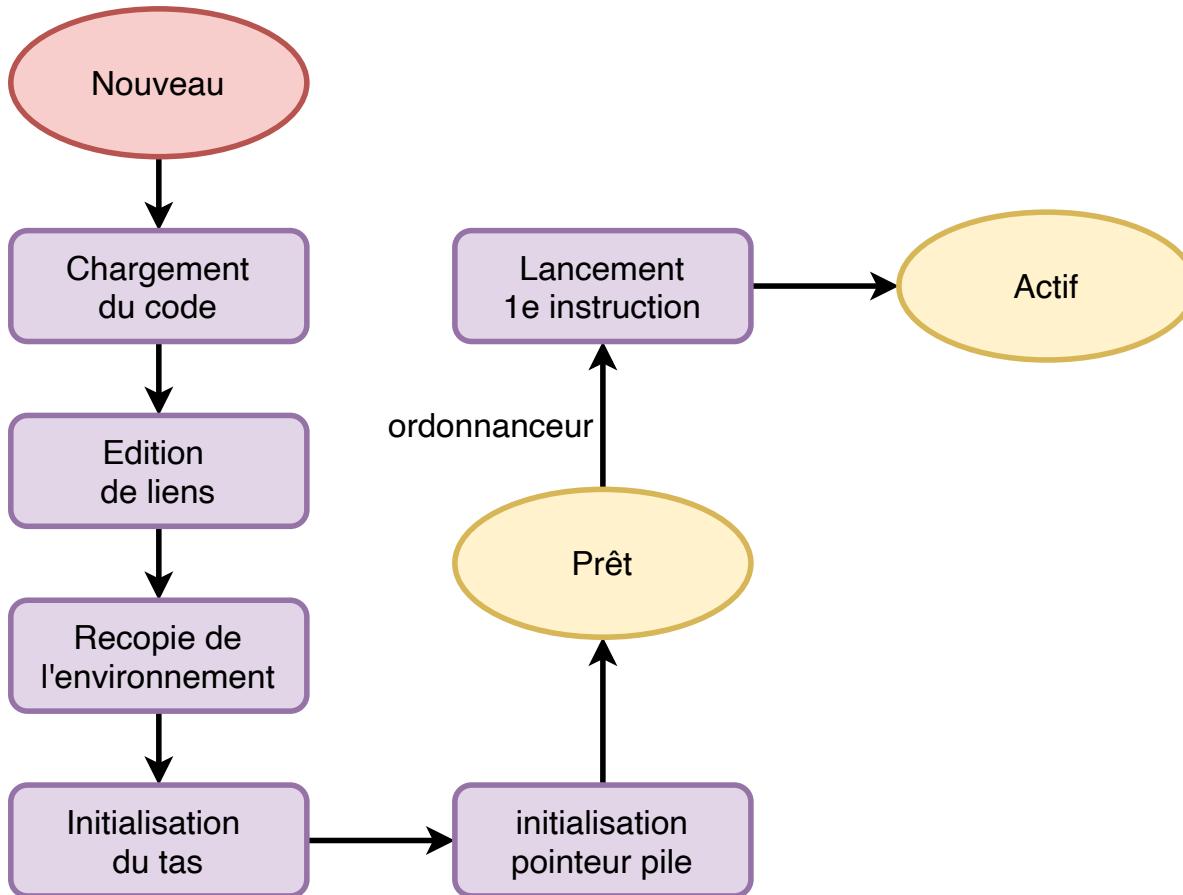


CYCLE DE VIE DU PROCESSUS



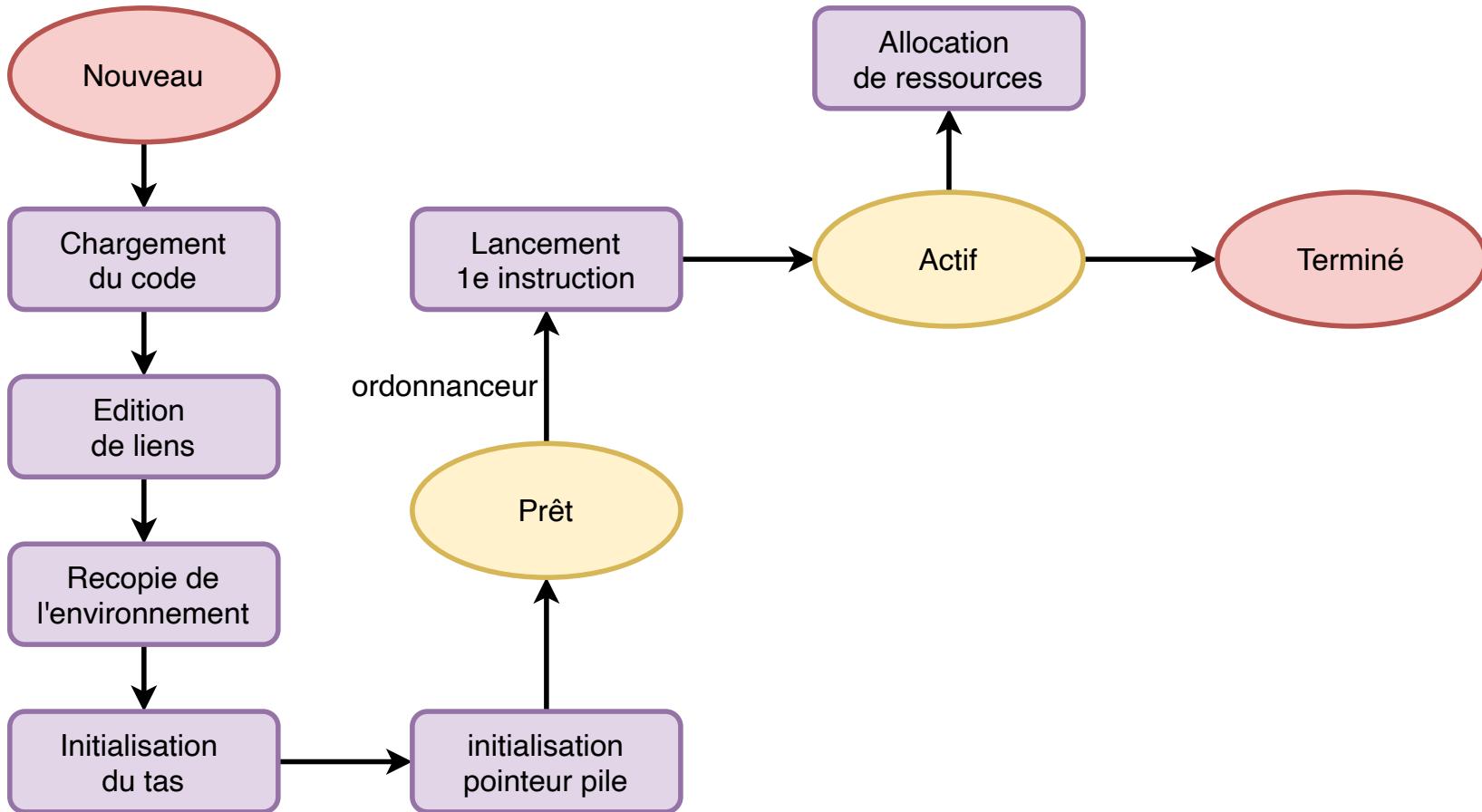
Le processus prêt est mis dans la file d'attente

CYCLE DE VIE DU PROCESSUS

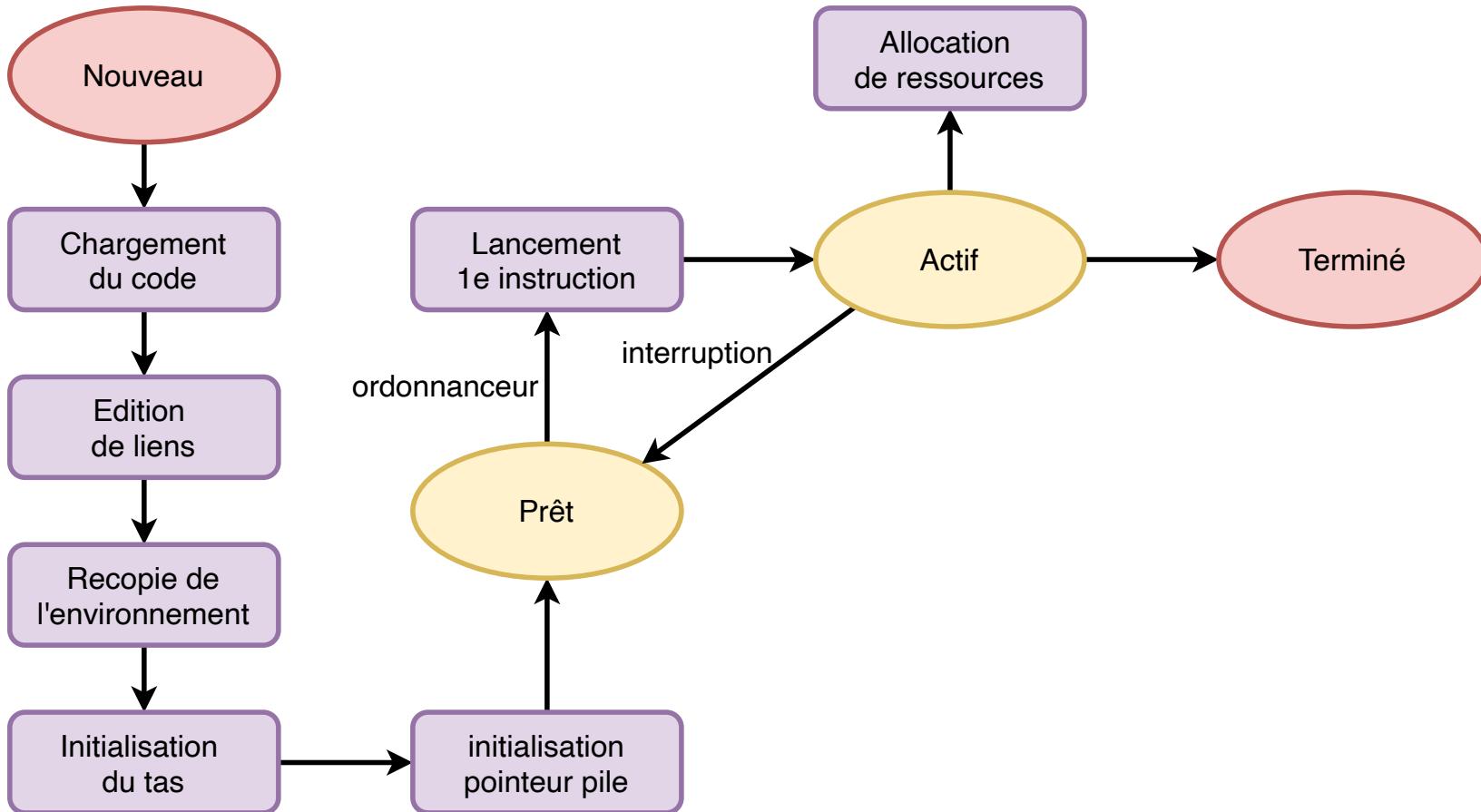


Le processus est sélectionné pour l'accès au processeur

CYCLE DE VIE DU PROCESSUS

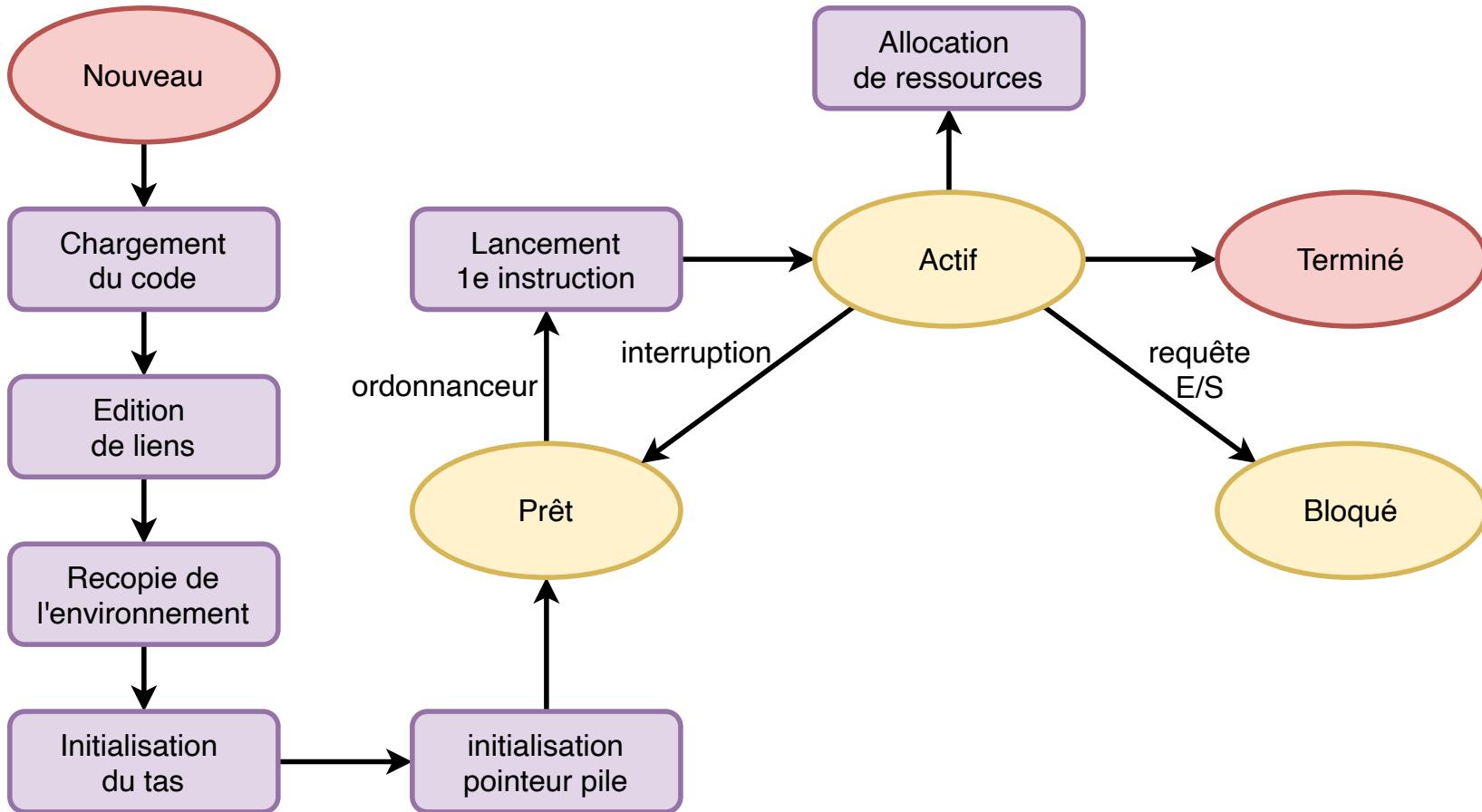


CYCLE DE VIE DU PROCESSUS



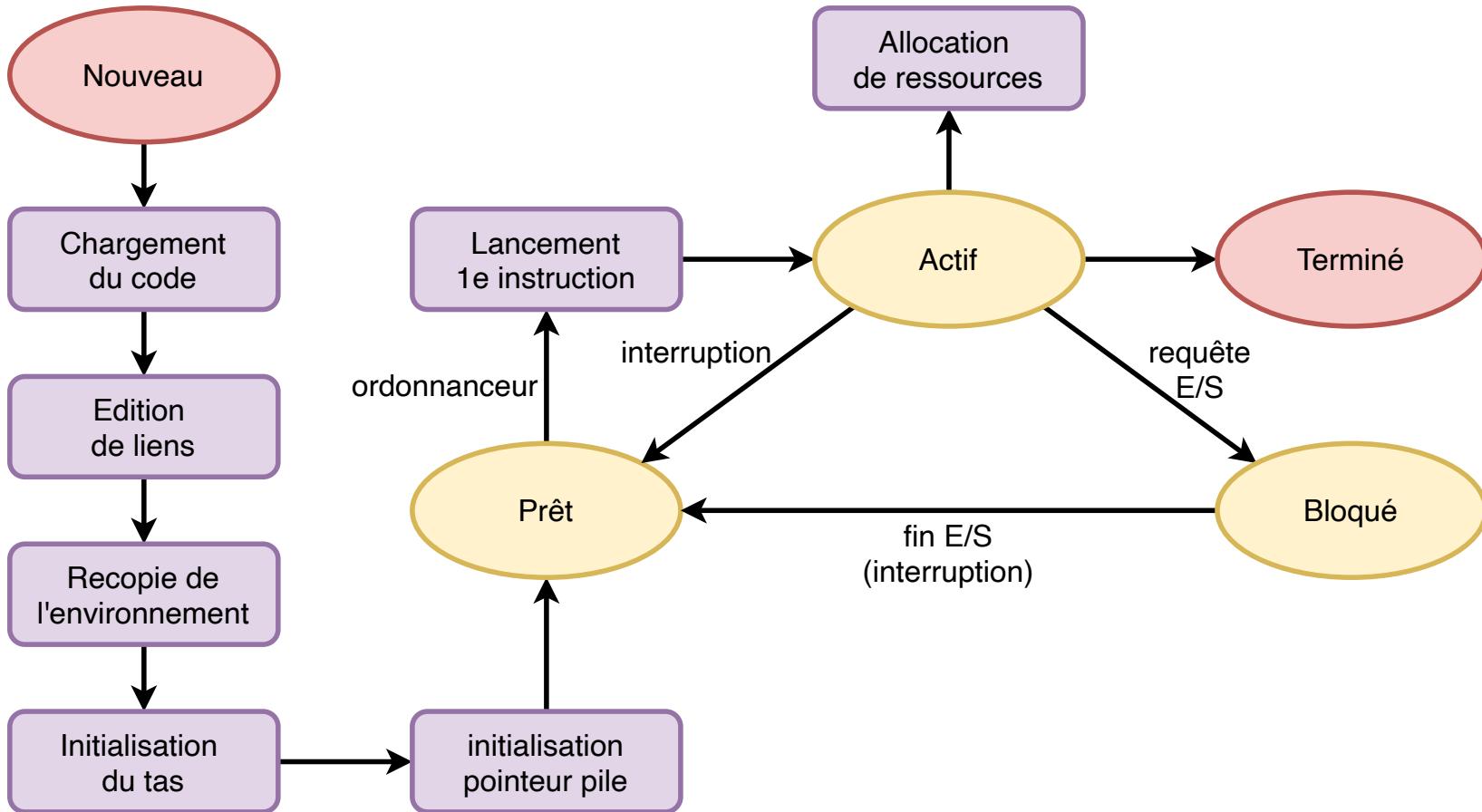
Un seul processus en exécution à la fois, mais plusieurs peuvent être prêts (file d'attente)

CYCLE DE VIE DU PROCESSUS

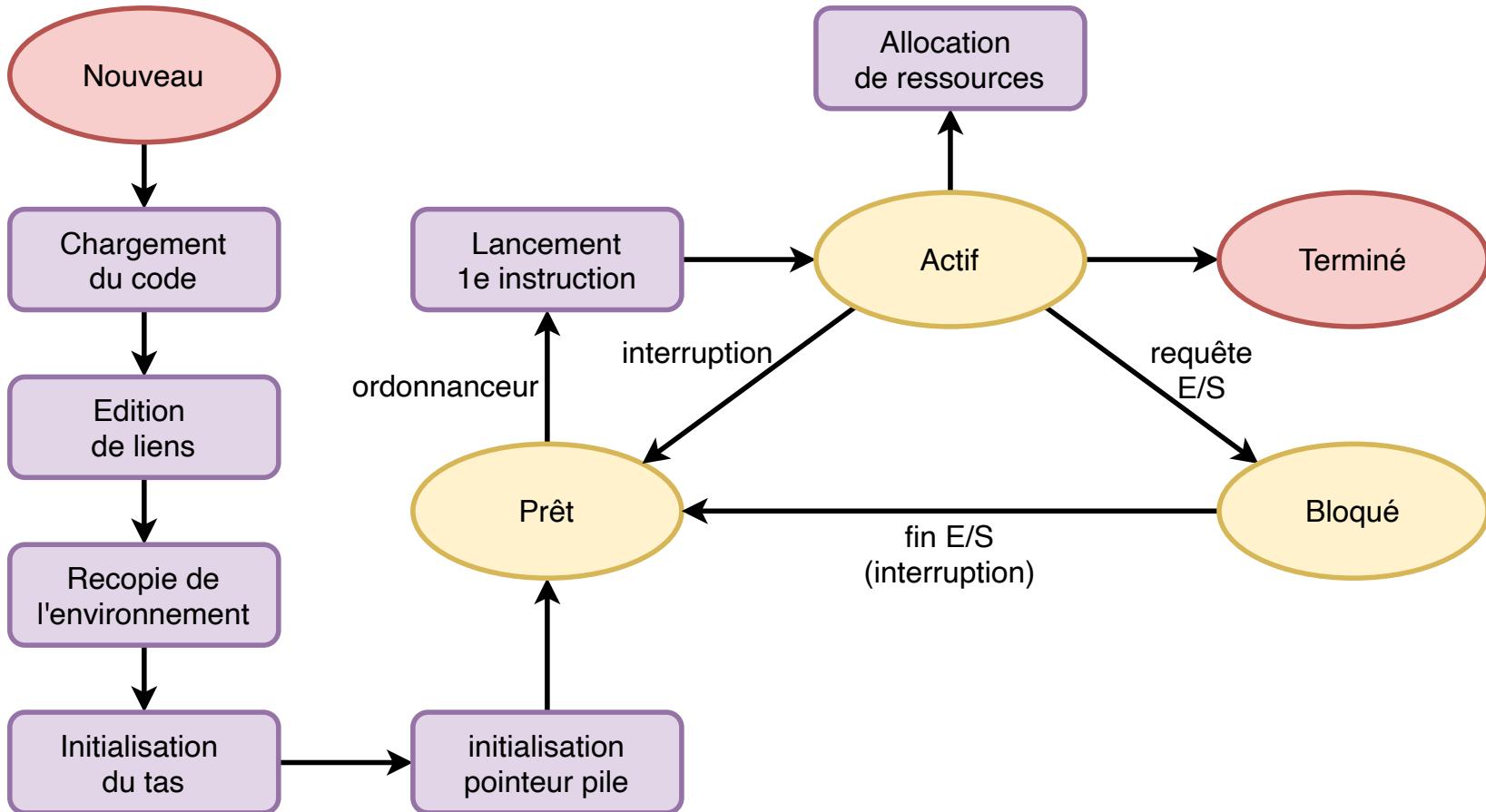


Les processus passent la main lorsqu'ils accèdent à une autre ressource (E/S)

CYCLE DE VIE DU PROCESSUS

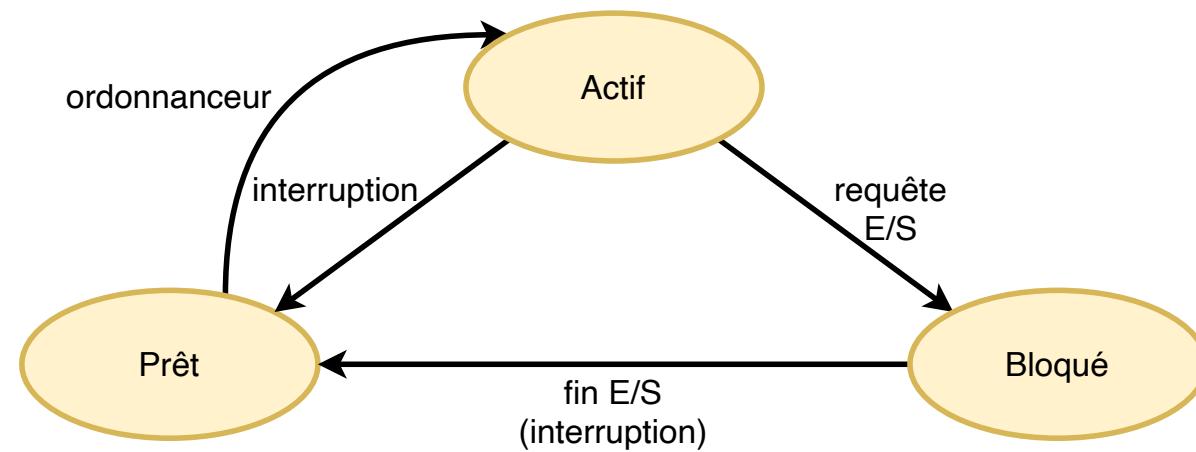


CYCLE DE VIE DU PROCESSUS



[0-1] processus en exécution, [0-n] processus prêts, [0-n] processus en attente

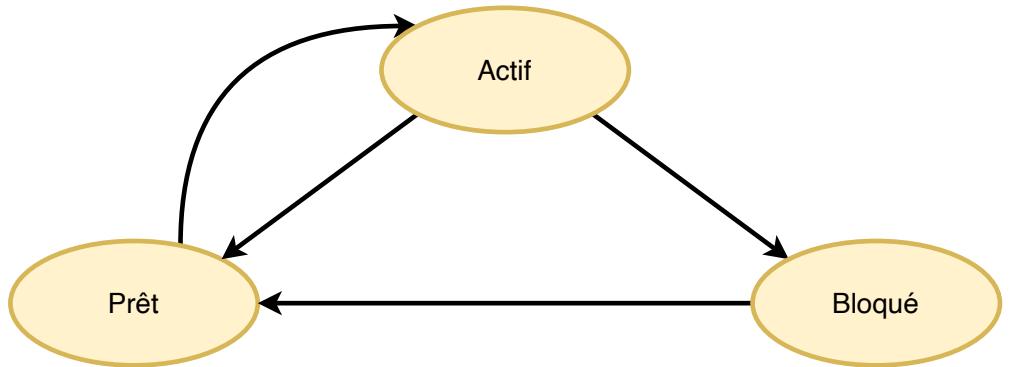
CYCLE DE VIE DU PROCESSUS



[0-1] processus en exécution, [0-n] processus prêts, [0-n] processus en attente

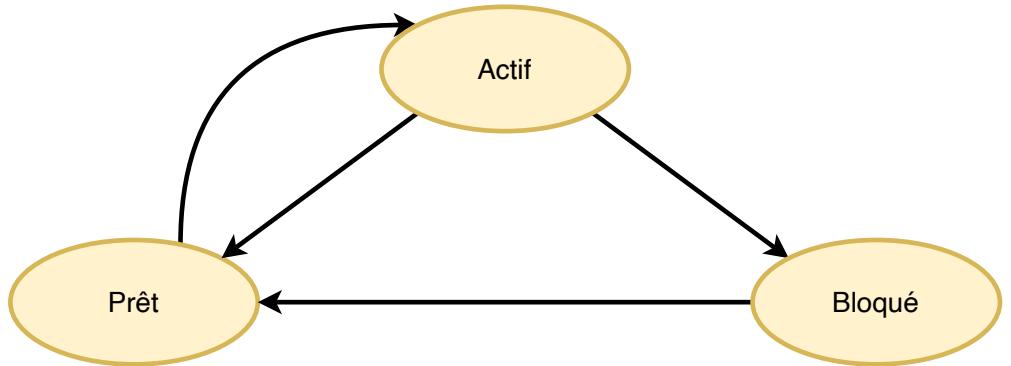
SUSPENSION DE L'EXÉCUTION

- Le processus en exécution laisse la main si:

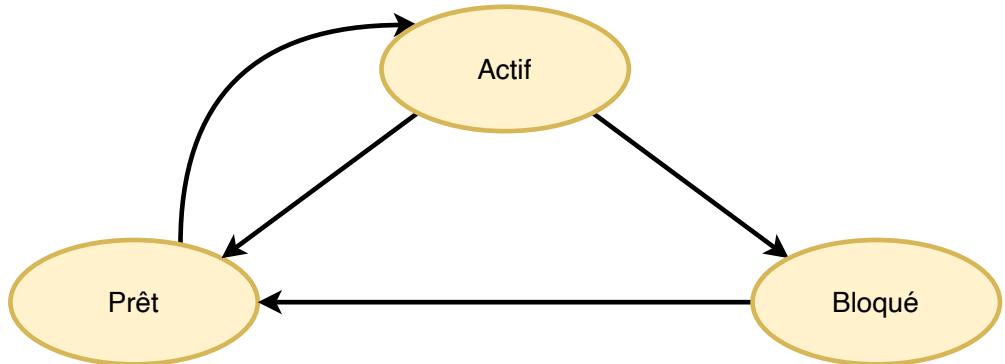


SUSPENSION DE L'EXÉCUTION

- Le processus en exécution laisse la main si:
👉 son quantum a expiré → **Prêt**

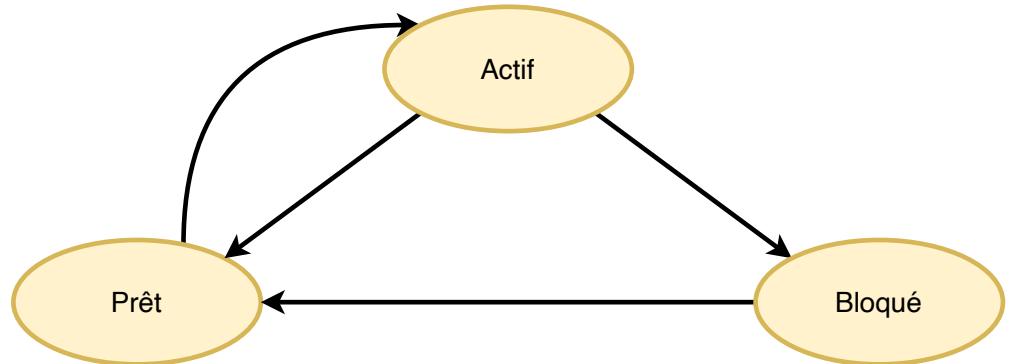


SUSPENSION DE L'EXÉCUTION



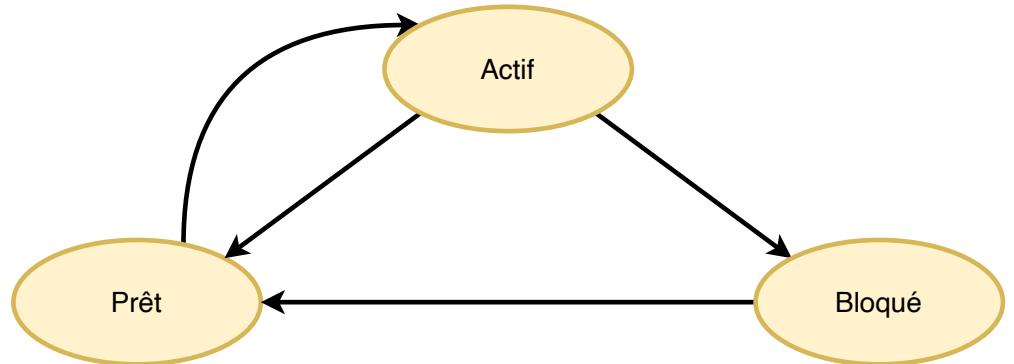
- Le processus en exécution laisse la main si:
 - 👉 son quantum a expiré → **Prêt**
 - 👉 crée un processus fils → **Prêt**

SUSPENSION DE L'EXÉCUTION



- Le processus en exécution laisse la main si:
 - 👉 son quantum a expiré → **Prêt**
 - 👉 crée un processus fils → **Prêt**
 - 👉 fait une demande d'E/S → **Bloqué**

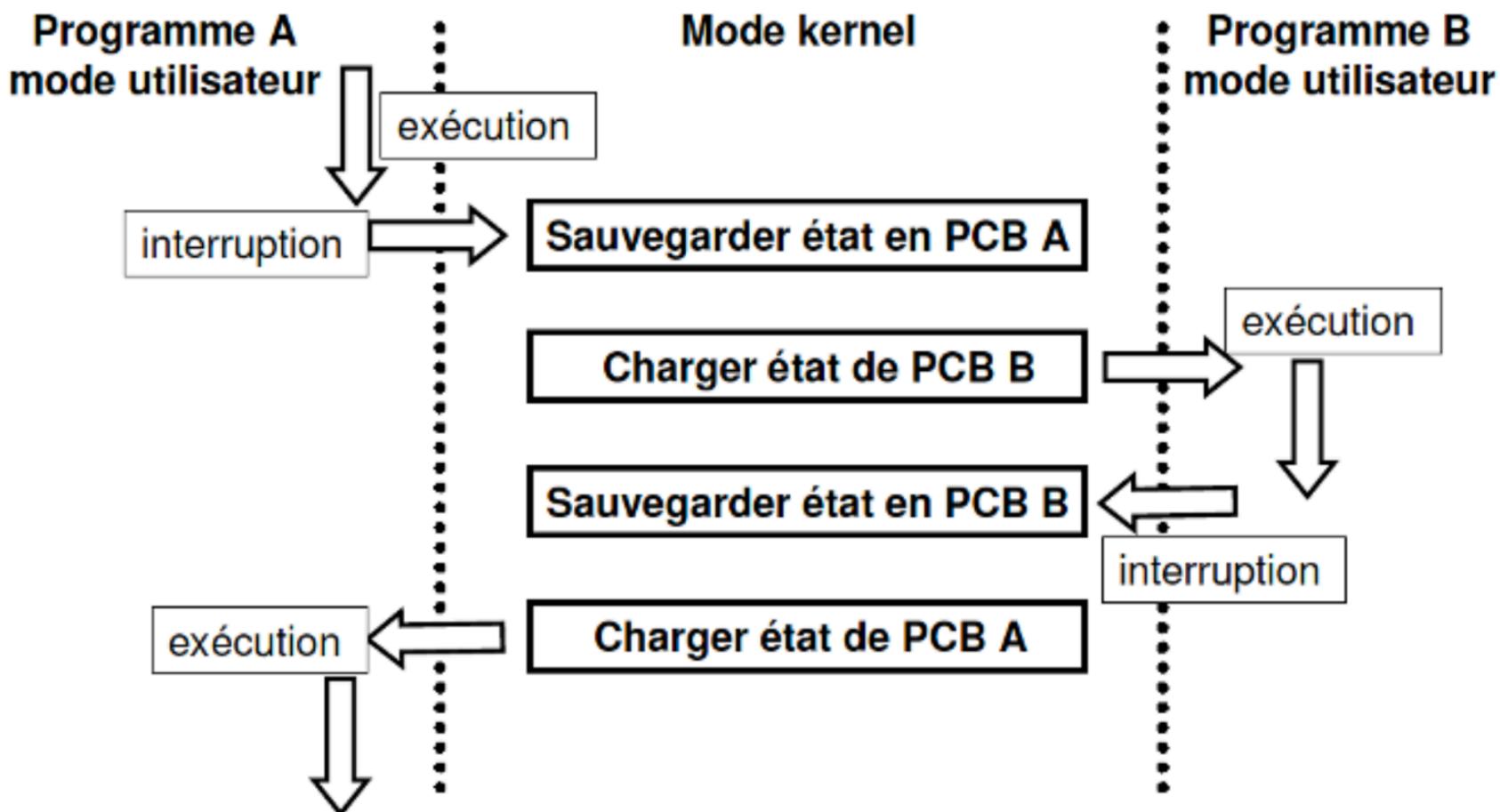
SUSPENSION DE L'EXÉCUTION



- Le processus en exécution laisse la main si:
 - 👉 son quantum a expiré → **Prêt**
 - 👉 crée un processus fils → **Prêt**
 - 👉 fait une demande d'E/S → **Bloqué**
 - 👉 exécute `wait` → **Bloqué**

COMMUTATION DE PROCESSUS

Changements de contexte



RÔLE DE L'OS

- **Création et suppression de processus**
 - Programme → processus
 - Munir le programme des **informations** nécessaires pour son **exécution**
- **Suspension et reprise**
 - Multiprogrammation → **interrompre et reprendre** les processus
 - **Gestion de la mémoire** où sont stockées les processus interrompus
- **Communication et synchronisation**
 - Partage de données entre plusieurs processus
 - **Consistance de l'état de la mémoire**

ACTIONS DE L'OS

ACTIONS DE L'OS

- **Mémoire:** chaque processus a son propre espace mémoire

ACTIONS DE L'OS

- **Mémoire:** chaque processus a son propre espace mémoire
👉 pas de problème de consistance mémoire/processeur

ACTIONS DE L'OS

- **Mémoire**: chaque processus a son propre espace mémoire
 - 👉 pas de problème de consistance mémoire/processeur
- **Verrous**: un processus peut verrouiller l'accès à une ressource

ACTIONS DE L'OS

- **Mémoire**: chaque processus a son propre espace mémoire
 - 👉 pas de problème de consistance mémoire/processeur
- **Verrous**: un processus peut verrouiller l'accès à une ressource
 - 👉 file d'attente pour l'accès à la ressource

ACTIONS DE L'OS

- **Mémoire**: chaque processus a son propre espace mémoire
 - 👉 pas de problème de consistance mémoire/processeur
- **Verrous**: un processus peut verrouiller l'accès à une ressource
 - 👉 file d'attente pour l'accès à la ressource
- Outils et Algorithmes de **synchronisation**

ACTIONS DE L'OS

- **Mémoire**: chaque processus a son propre espace mémoire
 - 👉 pas de problème de consistance mémoire/processeur
- **Verrous**: un processus peut verrouiller l'accès à une ressource
 - 👉 file d'attente pour l'accès à la ressource
- Outils et Algorithmes de **synchronisation**
 - 👉 voir section Synchronisation des processus

NOTION DE THREAD

NOTION DE THREAD

- **Un thread** est l'**unité d'exécution de base d'un processus**, décrite par son **point d'exécution** et son **état interne** (registres, pile, ...).

NOTION DE THREAD

- **Un thread** est l'**unité d'exécution de base d'un processus**, décrite par son **point d'exécution** et son **état interne** (registres, pile, ...).
 - les threads partagent le **même code et les mêmes données**

NOTION DE THREAD

- **Un thread** est l'**unité d'exécution de base d'un processus**, décrite par son **point d'exécution** et son **état interne** (registres, pile, ...).
 - les threads partagent le **même code et les mêmes données**
 - chaque thread a **sa propre pile**

NOTION DE THREAD

- **Un thread** est l'**unité d'exécution de base d'un processus**, décrite par son **point d'exécution** et son **état interne** (registres, pile, ...).
 - les threads partagent le **même code et les mêmes données**
 - chaque thread a **sa propre pile**
 - **un processus** peut avoir **plusieurs threads**

NOTION DE THREAD

- **Un thread** est l'**unité d'exécution de base d'un processus**, décrite par son **point d'exécution** et son **état interne** (registres, pile, ...).
 - les threads partagent le **même code et les mêmes données**
 - chaque thread a **sa propre pile**
 - **un processus** peut avoir **plusieurs threads**
- **Un thread** partage l'espace mémoire du processus qui l'a créé.

NOTION DE THREAD

- **Un thread** est l'**unité d'exécution de base d'un processus**, décrite par son **point d'exécution** et son **état interne** (registres, pile, ...).
 - les threads partagent le **même code et les mêmes données**
 - chaque thread a **sa propre pile**
 - **un processus** peut avoir **plusieurs threads**
- **Un thread** partage l'espace mémoire du processus qui l'a créé.
- **Un thread** est également appelé **processus léger**.

NOTION DE THREAD

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
👉 passage de ressources entre threads facilité.

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
 - 👉 passage de ressources entre threads facilité.
 - 👉 les variables sont dans le contexte du même processus.

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
 - 👉 passage de ressources entre threads facilité.
 - 👉 les variables sont dans le contexte du même processus.
- **Performances améliorées** par rapport aux processus :

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
 - 👉 passage de ressources entre threads facilité.
 - 👉 les variables sont dans le contexte du même processus.
- **Performances améliorées** par rapport aux processus :
 - création plus rapide;

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
 - 👉 passage de ressources entre threads facilité.
 - 👉 les variables sont dans le contexte du même processus.
- **Performances améliorées** par rapport aux processus :
 - création plus rapide;
 - changement de contexte plus rapide;

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
 - 👉 passage de ressources entre threads facilité.
 - 👉 les variables sont dans le contexte du même processus.
- **Performances améliorées** par rapport aux processus :
 - création plus rapide;
 - changement de contexte plus rapide;
 - partage du code → gain de place en mémoire;

UTILISATION DES THREADS

- Le **Thread** permet la gestion de **plusieurs traitements en parallèle** dans le même processus.
 - 👉 passage de ressources entre threads facilité.
 - 👉 les variables sont dans le contexte du même processus.
- **Performances améliorées** par rapport aux processus :
 - création plus rapide;
 - changement de contexte plus rapide;
 - partage du code → gain de place en mémoire;
 - réactivité → le processus s'exécute pendant qu'un thread est en attente.

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps
 - File d'attente de PCB

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps
 - File d'attente de PCB
 - Choisir un processus parmi tous les processus dans la file d'attente

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps
 - File d'attente de PCB
 - Choisir un processus parmi tous les processus dans la file d'attente
- Exemple

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps
 - File d'attente de PCB
 - Choisir un processus parmi tous les processus dans la file d'attente
- Exemple
 - le processeur est une ressource hautement critique.

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps
 - File d'attente de PCB
 - Choisir un processus parmi tous les processus dans la file d'attente
- Exemple
 - le processeur est une ressource hautement critique.
 - l'OS est en charge de sa répartition entre les processus

MULTIPROGRAMMATION ET TEMPS PARTAGÉ

- Les processus sont répartis sur les ressources :
 - Plusieurs processus peuvent vouloir la même ressource en même temps
 - File d'attente de PCB
 - Choisir un processus parmi tous les processus dans la file d'attente
- Exemple
 - le processeur est une ressource hautement critique.
 - l'OS est en charge de sa répartition entre les processus
 - 👉 l'ordonnancement (scheduling).

ORDONNANCEMENT

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 jusqu'à ce qu'il **termine**

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 - 👉 jusqu'à ce qu'il **termine**
 - 👉 jusqu'à ce qu'il **fasse une E/S**

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 - 👉 jusqu'à ce qu'il **termine**
 - 👉 jusqu'à ce qu'il **fasse une E/S**
 - 👉 jusqu'à ce que l'**OS** décide que **ce n'est plus son tour**

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 - 👉 jusqu'à ce qu'il **termine**
 - 👉 jusqu'à ce qu'il **fasse une E/S**
 - 👉 jusqu'à ce que l'**OS** décide que **ce n'est plus son tour**
- **Le remplacement d'un processus en exécution a un coût (commutation de contexte)**

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 - 👉 jusqu'à ce qu'il **termine**
 - 👉 jusqu'à ce qu'il **fasse une E/S**
 - 👉 jusqu'à ce que l'**OS** décide que **ce n'est plus son tour**
- **Le remplacement d'un processus en exécution a un coût (commutation de contexte)**
 - exécution de la routine d'ordonnancement

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 - 👉 jusqu'à ce qu'il **termine**
 - 👉 jusqu'à ce qu'il **fasse une E/S**
 - 👉 jusqu'à ce que l'**OS** décide que **ce n'est plus son tour**
- **Le remplacement d'un processus en exécution a un coût (commutation de contexte)**
 - exécution de la **routine d'ordonnancement**
 - sauvegarde du **contexte** (registres + PC)

ORDONNANCEMENT

- On ne s'intéresse pas à la durée totale du processus ...
mais au **temps** pendant lequel il va **garder le processeur** :
 - 👉 jusqu'à ce qu'il **termine**
 - 👉 jusqu'à ce qu'il **fasse une E/S**
 - 👉 jusqu'à ce que l'**OS** décide que **ce n'est plus son tour**
- **Le remplacement d'un processus en exécution a un coût (commutation de contexte)**
 - exécution de la **routine d'ordonnancement**
 - sauvegarde du **contexte** (registres + PC)
 - chargement d'un nouveau **contexte**

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être équitable (**fairness**) vis-à-vis des processus ;

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être **équitable** (**fairness**) vis-à-vis des processus ;
- maximiser l'utilisation globale du processeur (**efficace**) ;

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être **équitable** (**fairness**) vis-à-vis des processus ;
- maximiser l'utilisation globale du processeur (**efficace**) ;
- avoir un comportement le plus **prévisible** possible ;

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être **équitable** (**fairness**) vis-à-vis des processus ;
- maximiser l'utilisation globale du processeur (**efficace**) ;
- avoir un comportement le plus **prévisible** possible ;
- permettre un maximum d'utilisateurs interactifs (**réactif**) ;

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être **équitable** (**fairness**) vis-à-vis des processus ;
- maximiser l'utilisation globale du processeur (**efficace**) ;
- avoir un comportement le plus **prévisible** possible ;
- permettre un maximum d'utilisateurs interactifs (**réactif**) ;
- **minimiser le surcoût** (**overhead**) lier à la parallélisation ;

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être **équitable** (**fairness**) vis-à-vis des processus ;
- maximiser l'utilisation globale du processeur (**efficace**) ;
- avoir un comportement le plus **prévisible** possible ;
- permettre un maximum d'utilisateurs interactifs (**réactif**) ;
- **minimiser le surcoût** (**overhead**) lier à la parallélisation ;
- assurer une **utilisation maximale** des ressources ;

OBJECTIFS POSSIBLES DE L'ORDONNANCEMENT

- être **équitable** (**fairness**) vis-à-vis des processus ;
- maximiser l'utilisation globale du processeur (**efficace**) ;
- avoir un comportement le plus **prévisible** possible ;
- permettre un maximum d'utilisateurs interactifs (**réactif**) ;
- **minimiser le surcoût** (**overhead**) lier à la parallélisation ;
- assurer une **utilisation maximale** des ressources ;
- gérer convenablement les **priorités**.

ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

- Ordonnancement **non préemptif** → après avoir donné le contrôle à un processus, l'OS **ne peut pas l'interrompre**

ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

- Ordonnancement **non préemptif** → après avoir donné le contrôle à un processus, l'OS **ne peut pas l'interrompre**
 - sauf si en attente d'une ressource

ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

- Ordonnancement **non préemptif** → après avoir donné le contrôle à un processus, l'OS **ne peut pas l'interrompre**
 - sauf si en attente d'une ressource
- Ordonnancement **préemptif (avec réquisition)** → l'OS **peut interrompre un processus si :**

ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

- Ordonnancement **non préemptif** → après avoir donné le contrôle à un processus, l'OS **ne peut pas l'interrompre**
 - sauf si en attente d'une ressource
- Ordonnancement **préemptif (avec réquisition)** → l'OS peut interrompre un processus si :
 - 👉 le **quantum** (le temps d'utilisation maximum consécutif) est atteint

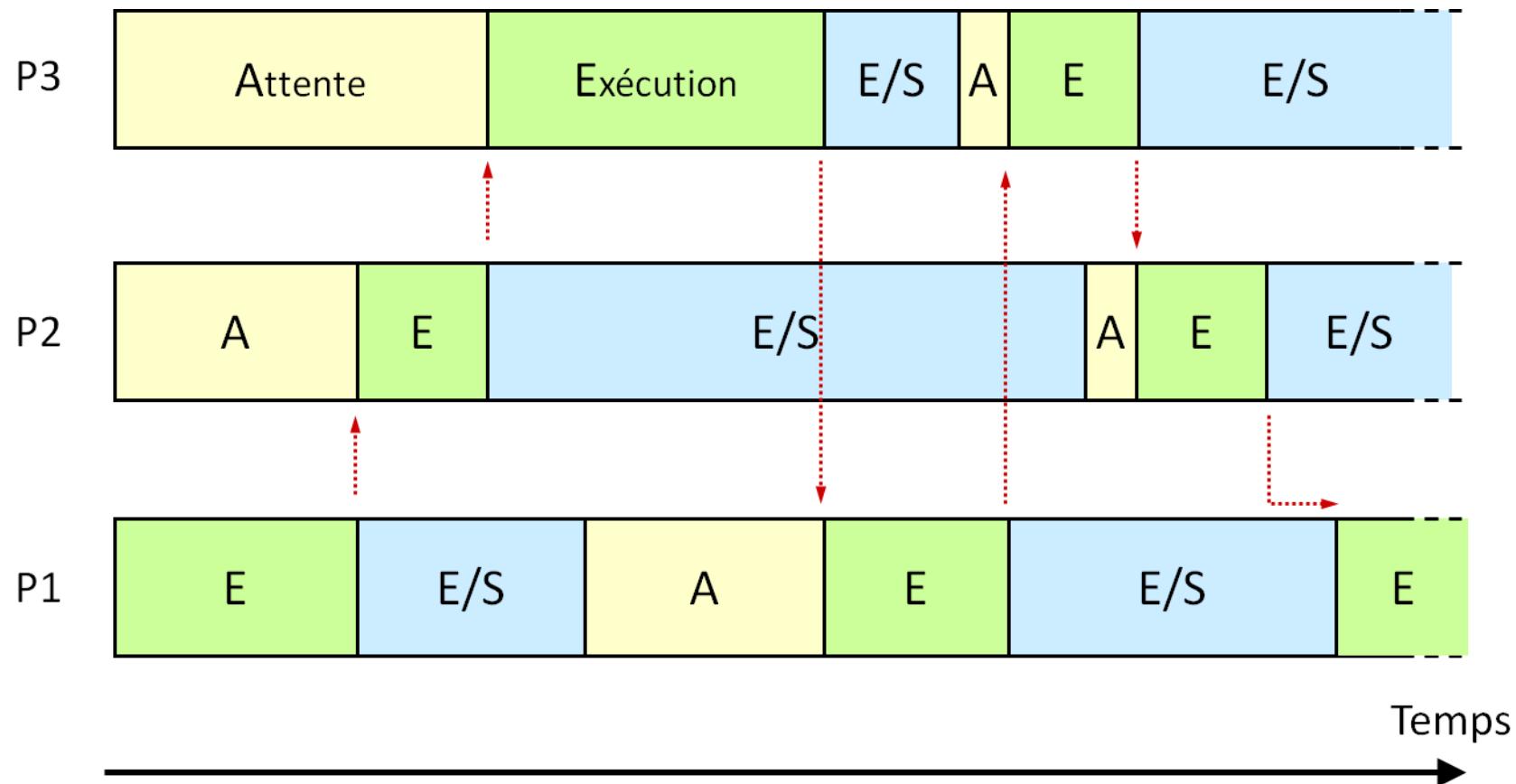
ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

- Ordonnancement **non préemptif** → après avoir donné le contrôle à un processus, l'OS **ne peut pas l'interrompre**
 - sauf si en attente d'une ressource
- Ordonnancement **préemptif (avec réquisition)** → l'OS peut interrompre un processus si :
 - 👉 le **quantum** (le temps d'utilisation maximum consécutif) est atteint
 - 👉 **un processus plus prioritaire** demande d'utiliser le processeur

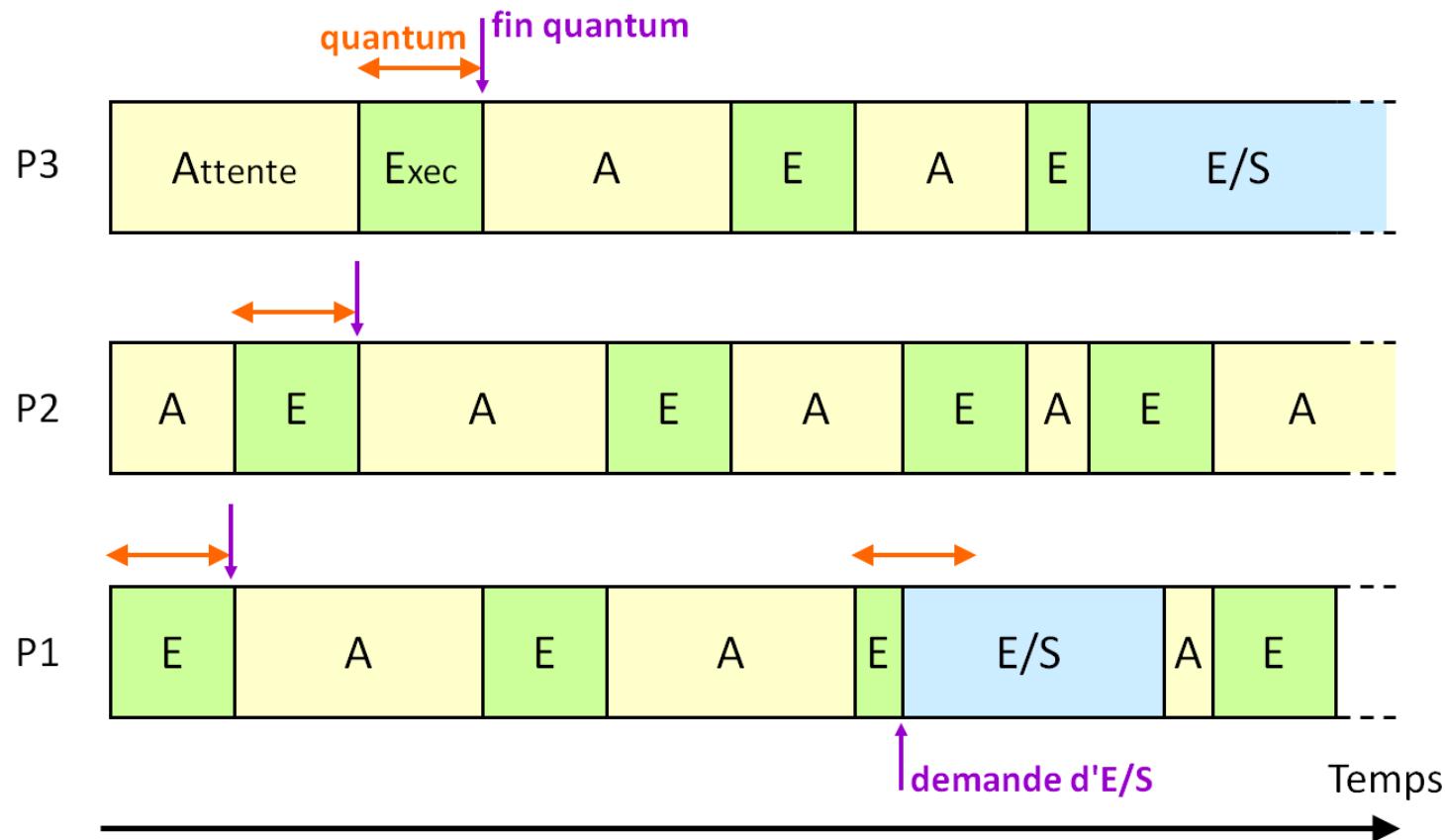
ORDONNANCEMENT PRÉEMPTIF vs NON PRÉEMPTIF

- Ordonnancement **non préemptif** → après avoir donné le contrôle à un processus, l'OS **ne peut pas l'interrompre**
 - sauf si en attente d'une ressource
- Ordonnancement **préemptif (avec réquisition)** → l'OS peut interrompre un processus si :
 - 👉 le **quantum** (le temps d'utilisation maximum consécutif) est atteint
 - 👉 **un processus plus prioritaire** demande d'utiliser le processeur
- **L'ordonnancement préemptif** est indispensable pour gérer des **systèmes temps réel** ou des **systèmes interactifs**.

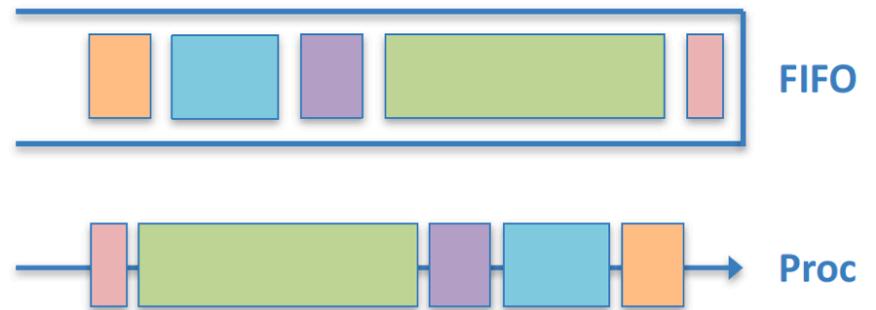
ORDONNANCEMENT NON PRÉEMPTIF



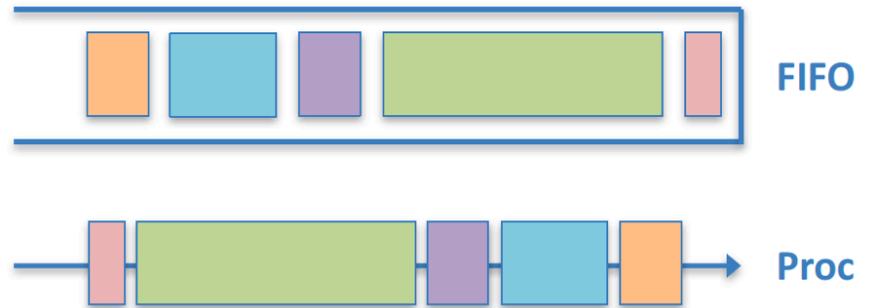
ORDONNANCEMENT PRÉEMPTIF



DE NOMBREUSES STRATÉGIES

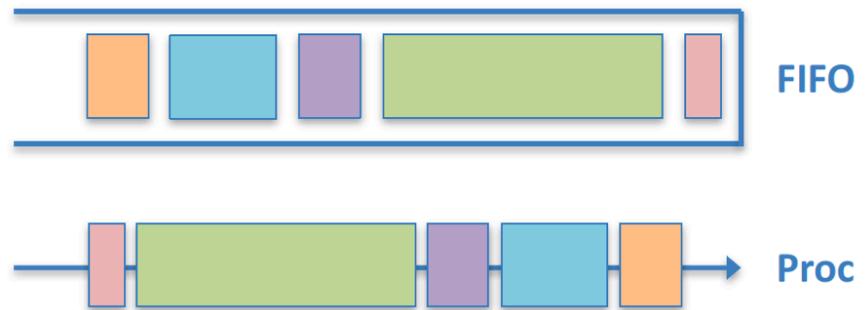


DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION



DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**
👉 premier arrivé, premier servi.



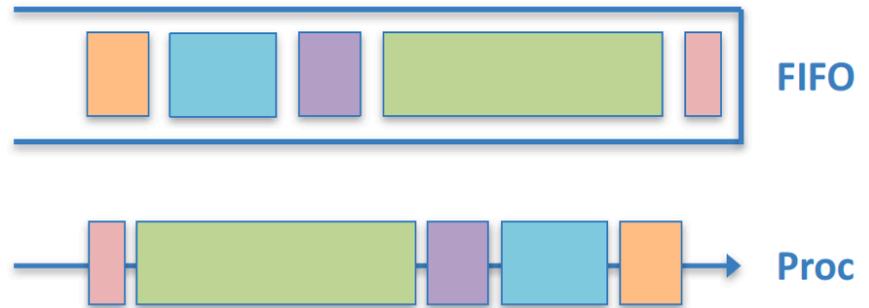
DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**
👉 premier arrivé, premier servi.
- **Avantages :**



DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**
👉 premier arrivé, premier servi.
- **Avantages :**
✓ Simple à implémenter



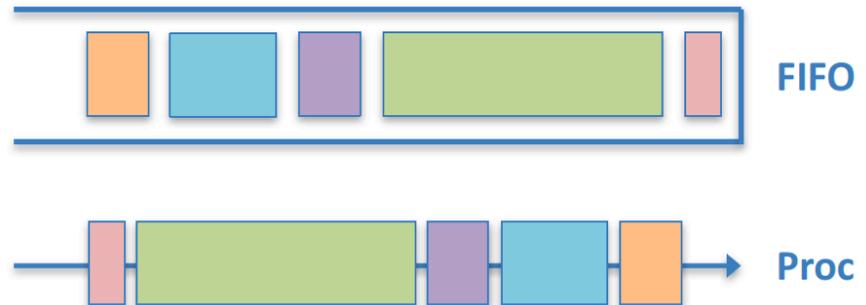
DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**
👉 premier arrivé, premier servi.
- **Avantages :**
 - ✓ Simple à implémenter
 - ✓ Équitable dans l'ordre d'arrivée



DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**
👉 premier arrivé, premier servi.
- **Avantages :**
 - ✓ Simple à implémenter
 - ✓ Équitable dans l'ordre d'arrivée
- **Inconvénients :**



DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**

👉 premier arrivé, premier servi.



- **Avantages :**

- ✓ Simple à implémenter
- ✓ Équitable dans l'ordre d'arrivée

- **Inconvénients :**

- ✗ Peu efficace → des processus ont "longtemps" le processeur

DE NOMBREUSES STRATÉGIES FIFO SANS PRÉEMPTION

- **Principe:**

👉 premier arrivé, premier servi.



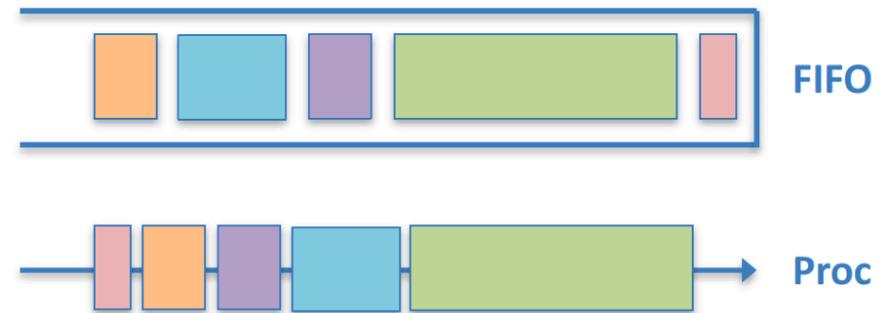
- **Avantages :**

- ✓ Simple à implémenter
- ✓ Équitable dans l'ordre d'arrivée

- **Inconvénients :**

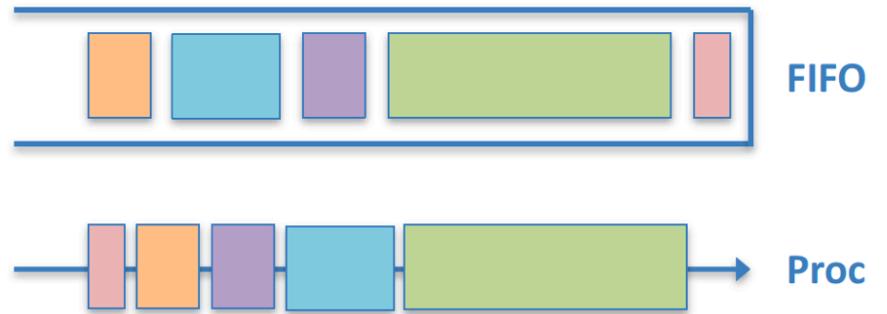
- ✗ Peu efficace → des processus ont "longtemps" le processeur
- ✗ Peu réactif → des processus peuvent attendre longtemps

DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD



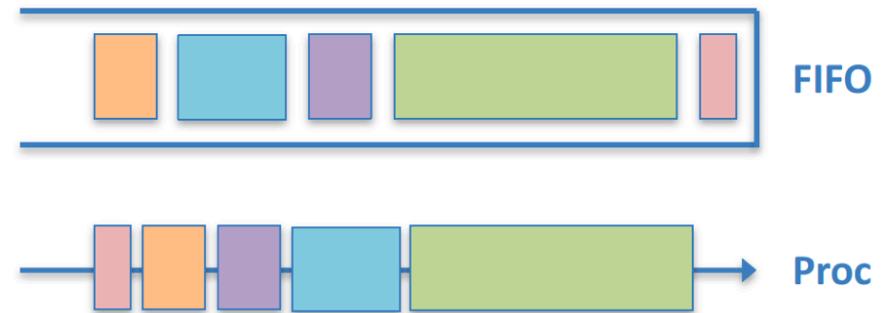
DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**
👉 priorité au processus le plus court



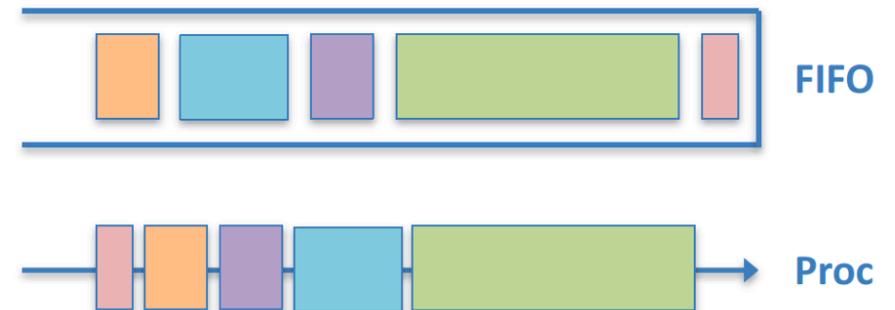
DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**
👉 priorité au processus le plus court
- **Avantages :**



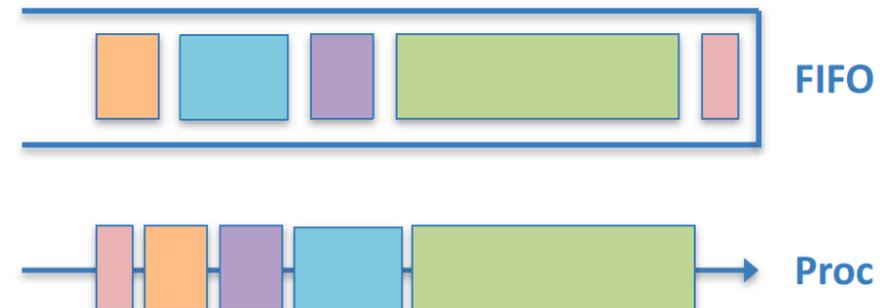
DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**
👉 priorité au processus le plus court
- **Avantages :**
✓ Réactif: avantage aux petits processus



DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**
👉 priorité au processus le plus court
- **Avantages :**
 - ✓ Réactif: avantage aux petits processus
 - ✓ Optimal sur le temps d'attente moyen



DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**
👉 priorité au processus le plus court
- **Avantages :**
 - ✓ Réactif: avantage aux petits processus
 - ✓ Optimal sur le temps d'attente moyen
- **Inconvénients :**



DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**
👉 priorité au processus le plus court
 - **Avantages :**
 - ✓ Réactif: avantage aux petits processus
 - ✓ Optimal sur le temps d'attente moyen
 - **Inconvénients :**
 - ✗ Pas efficace: les processus ont moins le processeur s'il y a beaucoup d'E/S
- 

DE NOMBREUSES STRATÉGIES PLUS COURT D'ABORD

- **Principe:**

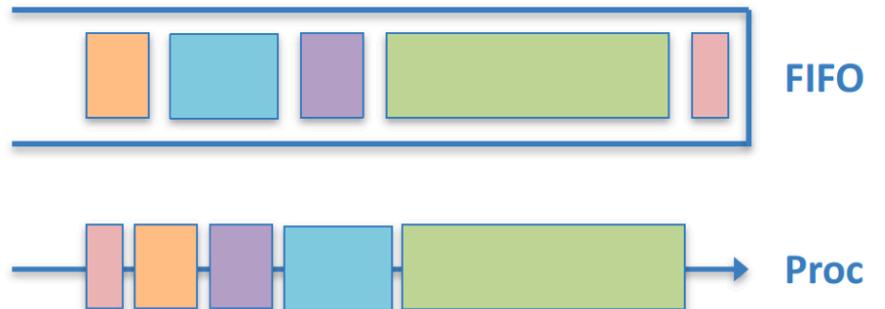
👉 priorité au processus le plus court

- **Avantages :**

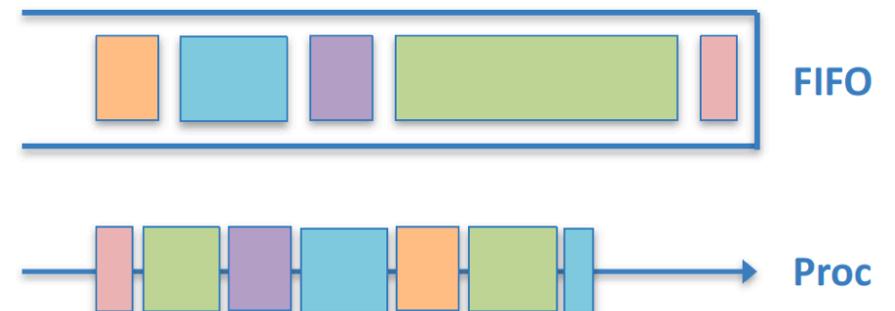
- ✓ Réactif: avantage aux petits processus
- ✓ Optimal sur le temps d'attente moyen

- **Inconvénients :**

- ✗ Pas efficace: les processus ont moins le processeur s'il y a beaucoup d'E/S
- ✗ Non équitable: on peut avoir une **famine** des gros processus

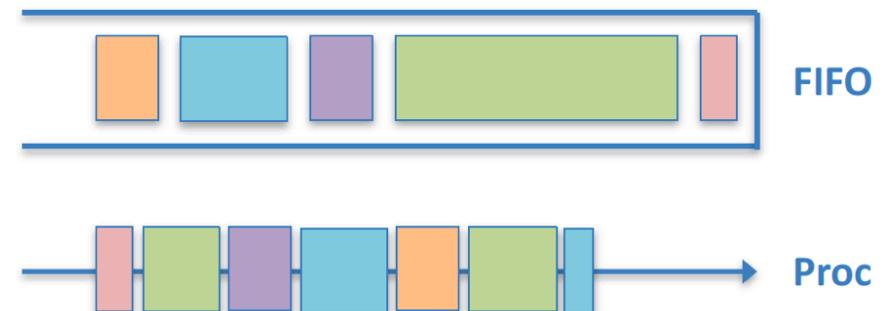


DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)



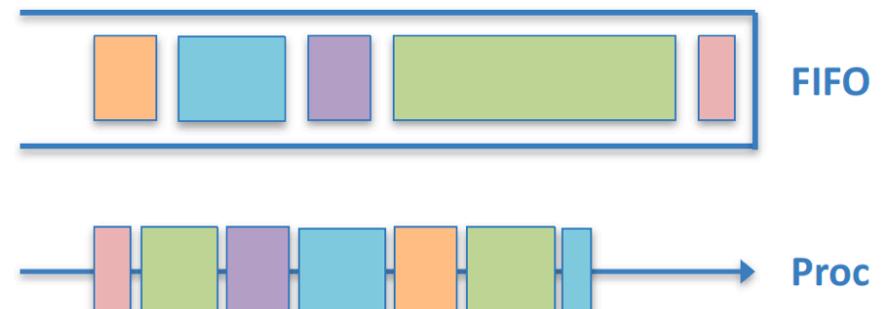
DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec **quantum** de temps



DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec **quantum** de temps
- **Avantages :**

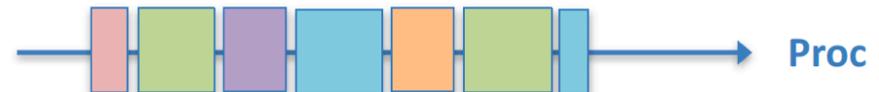


DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec **quantum** de temps

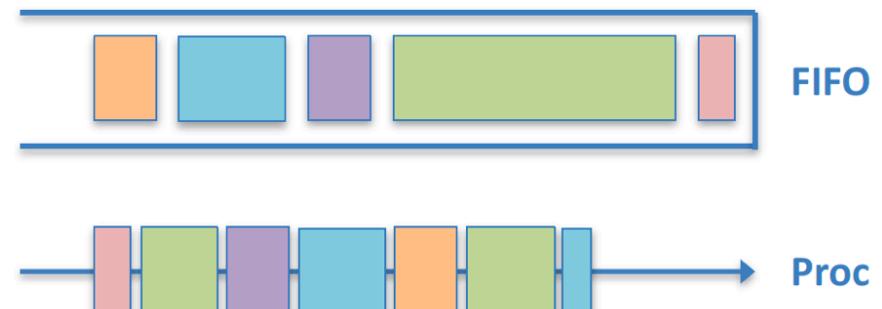


- **Avantages :**
✓ Équitable: tout le monde a le processeur



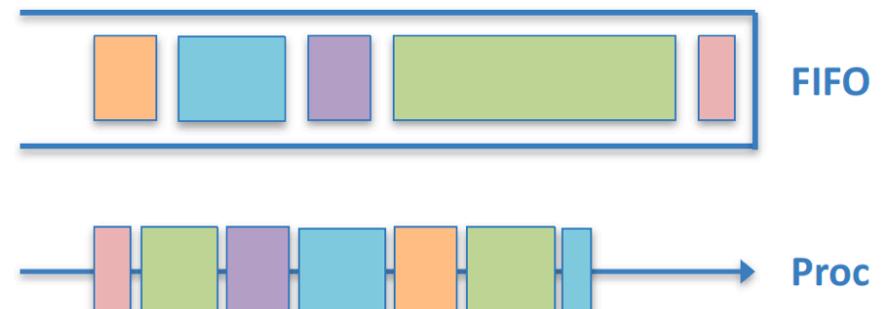
DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec **quantum** de temps
- **Avantages :**
 - ✓ Équitable: tout le monde a le processeur
 - ✓ Réactif: les processus n'attendent pas



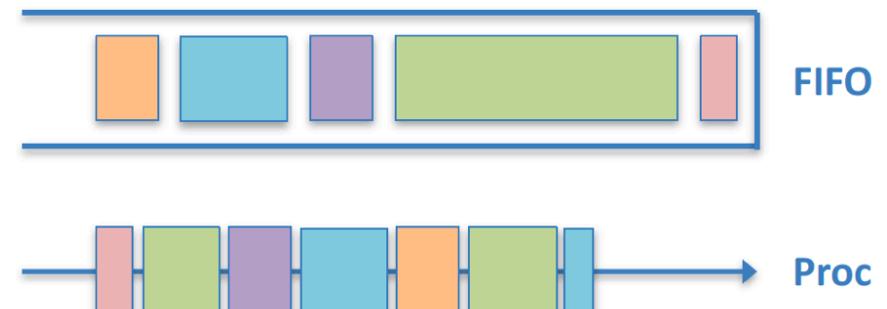
DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec **quantum** de temps
- **Avantages :**
 - ✓ Équitable: tout le monde a le processeur
 - ✓ Réactif: les processus n'attendent pas
- **Inconvénients :**



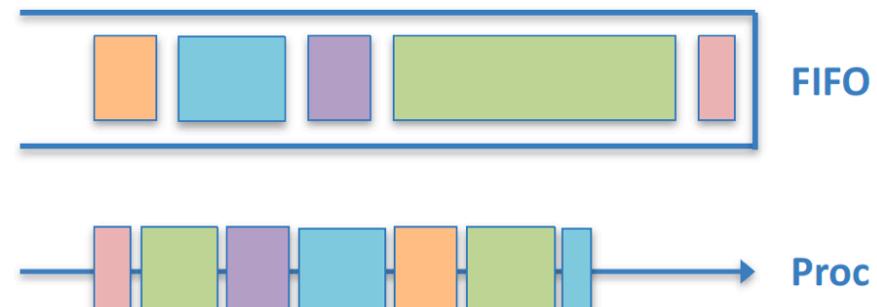
DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec quantum de temps
- **Avantages :**
 - ✓ Équitable: tout le monde a le processeur
 - ✓ Réactif: les processus n'attendent pas
- **Inconvénients :**
 - ✗ Temps d'attente moyen plus élevé



DE NOMBREUSES STRATÉGIES ROUND-ROBIN (FIFO PRÉEMPTIF)

- **Principe:**
👉 FIFO avec quantum de temps



- **Avantages :**
 - ✓ Équitable: tout le monde a le processeur
 - ✓ Réactif: les processus n'attendent pas

- **Inconvénients :**
 - ✗ Temps d'attente moyen plus élevé
 - ✗ Beaucoup de commutations → surcoût!

DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS



DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS

- Plusieurs files d'attente



DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS

- Plusieurs files d'attente
- La durée des quantums peut dépendre de la file



DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS

- Plusieurs files d'attente
- La durée des quantums peut dépendre de la file
- ✓ favorise le temps de réponse des processus systèmes

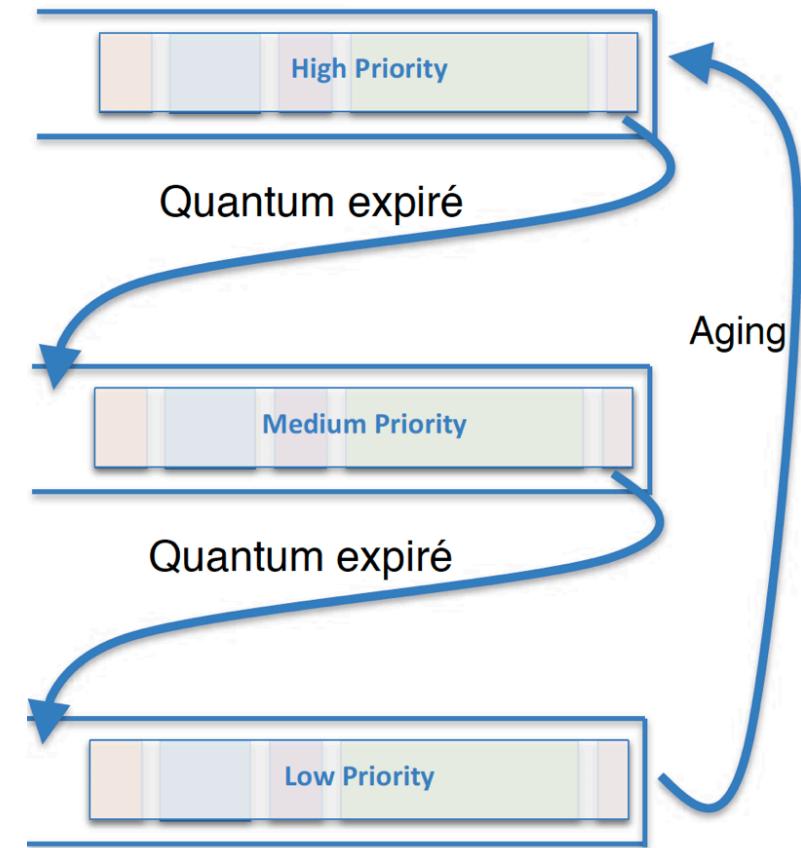


DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS

- Plusieurs files d'attente
 - La durée des quantums peut dépendre de la file
- ✓ favorise le temps de réponse des processus systèmes
- ✗ Comment choisir la priorité ?

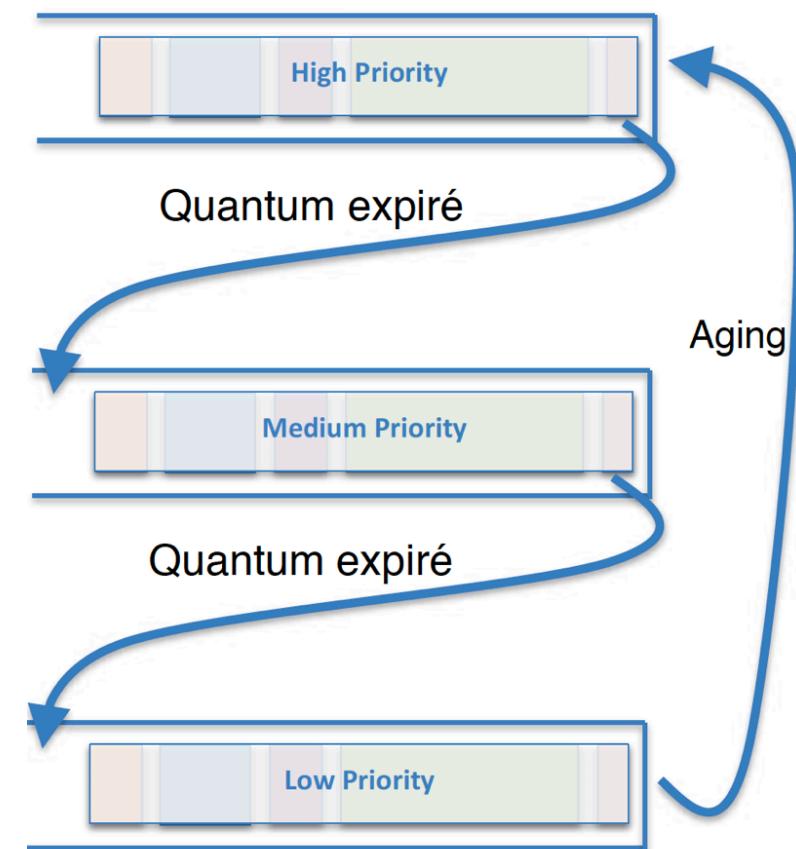


DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS DYNAMIQUES



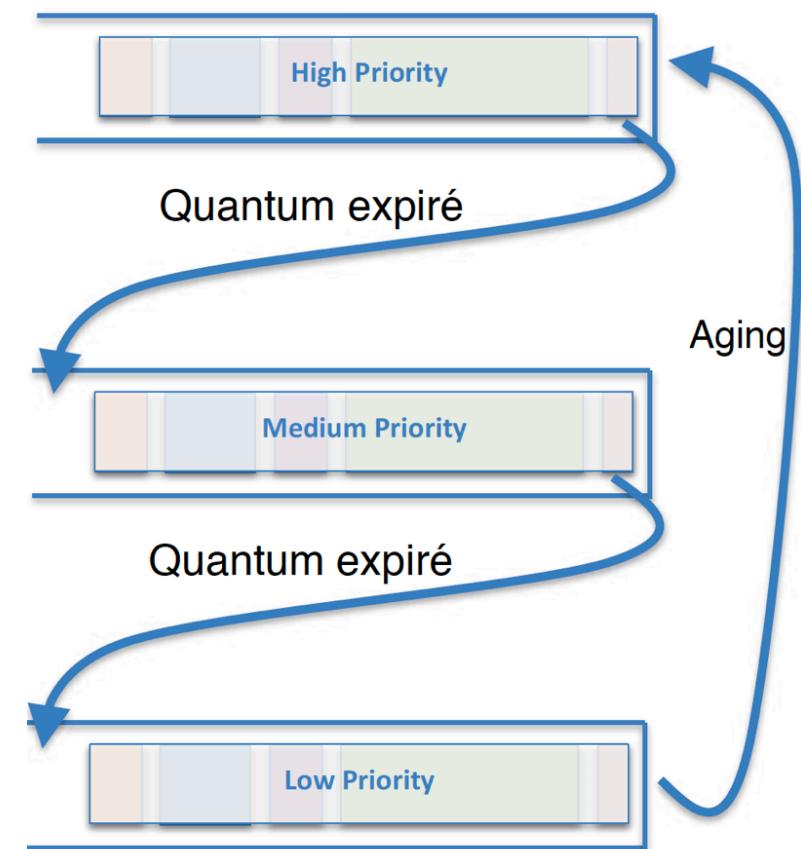
DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS DYNAMIQUES

- La priorité d'un processus peut être modifiée par l'OS



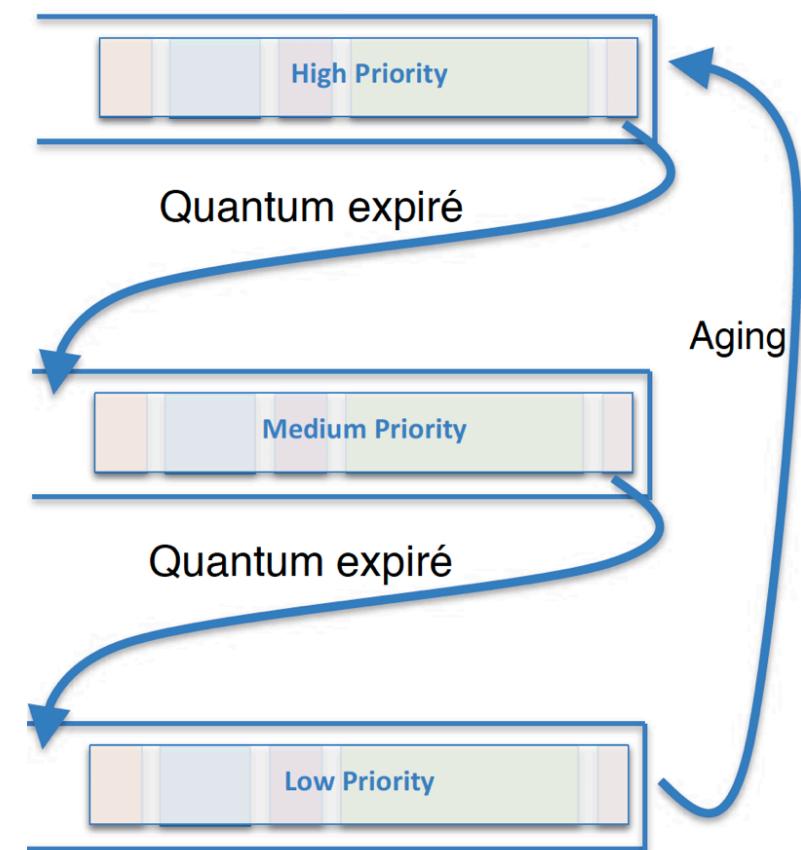
DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS DYNAMIQUES

- La priorité d'un processus peut être modifiée par l'OS
 - Par exemple sortie d'attente I/O



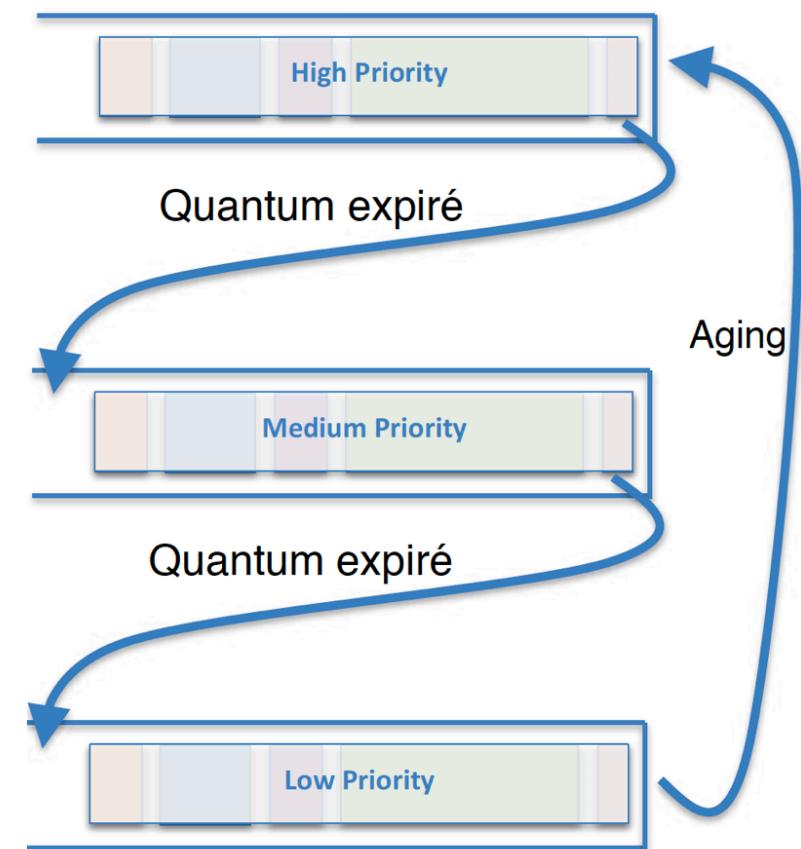
DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS DYNAMIQUES

- La priorité d'un processus peut être modifiée par l'OS
 - Par exemple sortie d'attente I/O
 - Attente longue en file basse priorité



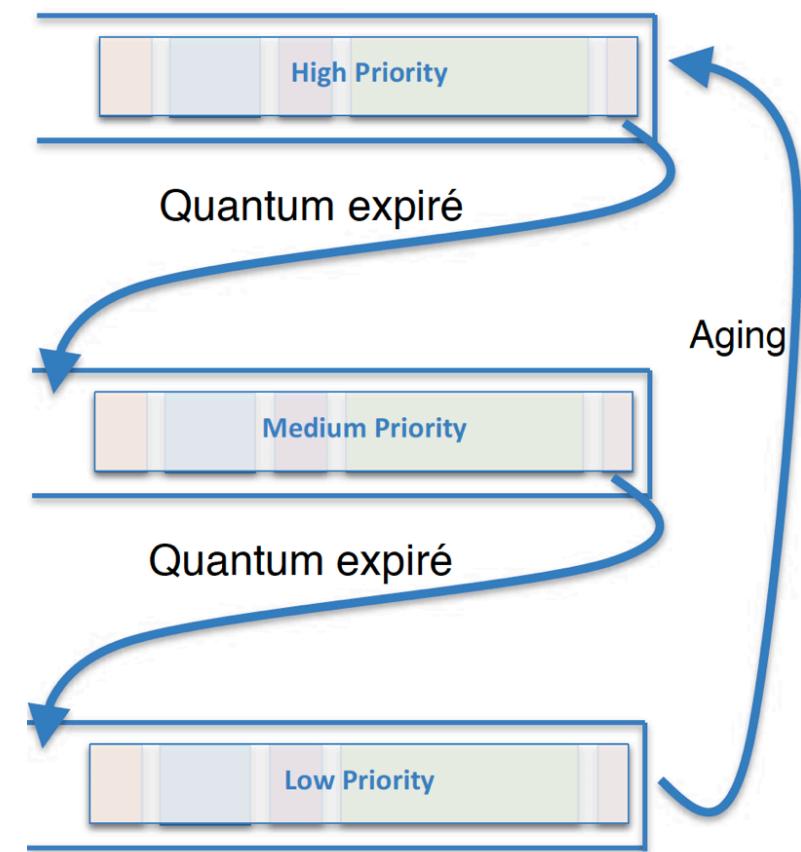
DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS DYNAMIQUES

- La priorité d'un processus peut être modifiée par l'OS
 - Par exemple sortie d'attente I/O
 - Attente longue en file basse priorité
- **Utilité** : Windows NT, OS X, Linux



DE NOMBREUSES STRATÉGIES FILES DE PRIORITÉS DYNAMIQUES

- La priorité d'un processus peut être modifiée par l'OS
 - Par exemple sortie d'attente I/O
 - Attente longue en file basse priorité
- Utilité : Windows NT, OS X, Linux
- Notion de **classe de priorités**



EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

- **Principe**

EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

- **Principe**
 - Quantum de temps selon priorité (0 = priorité max)

EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

- **Principe**

- Quantum de temps selon priorité (0 = priorité max)
- Priorité modifiée à la fin du quantum ou après une E/S

EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

- **Principe**

- Quantum de temps selon priorité (0 = priorité max)
- Priorité modifiée à la fin du quantum ou après une E/S
 - 👉 Prioritaire → grand quantum

EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

- **Principe**

- Quantum de temps selon priorité (0 = priorité max)
- Priorité modifiée à la fin du quantum ou après une E/S
 - 👉 Prioritaire → grand quantum
 - 👉 Quantum consommé → priorité augmentée

EXAMPLE : SOLARIS

OS des machines Sun entre 1993 et 2000

- **Principe**

- Quantum de temps selon priorité (0 = priorité max)
- Priorité modifiée à la fin du quantum ou après une E/S
 - 👉 Prioritaire → grand quantum
 - 👉 Quantum consommé → priorité augmentée
 - 👉 E/S → priorité diminuée

EXAMPLE : WINDOWS XP ET APRÈS

EXAMPLE : WINDOWS XP ET APRÈS

- Principe

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**
 - Priorité + Round Robin

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**
 - Priorité + Round Robin
 - 👉 Gérée au niveau des threads uniquement

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**
 - Priorité + Round Robin
 - 👉 Gérée au niveau des threads uniquement
 - 👉 32 niveaux de priorité

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**

- Priorité + Round Robin

- 👉 Gérée au niveau des threads uniquement

- 👉 32 niveaux de priorité

- 👉 Ordonnancement préemptif par niveau de priorité

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**
 - Priorité + Round Robin
 - 👉 Gérée au niveau des threads uniquement
 - 👉 32 niveaux de priorité
 - 👉 Ordonnancement préemptif par niveau de priorité
 - Priorité dynamique

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**
 - Priorité + Round Robin
 - 👉 Gérée au niveau des threads uniquement
 - 👉 32 niveaux de priorité
 - 👉 Ordonnancement préemptif par niveau de priorité
 - Priorité dynamique
 - 👉 Baissée à la fin du quantum

EXAMPLE : WINDOWS XP ET APRÈS

- **Principe**

- Priorité + Round Robin

- 👉 Gérée au niveau des threads uniquement

- 👉 32 niveaux de priorité

- 👉 Ordonnancement préemptif par niveau de priorité

- Priorité dynamique

- 👉 Baissée à la fin du quantum

- 👉 Remontée après chaque E/S → interface graphique plus réactive!

EXAMPLE : LINUX, MACOS X

EXAMPLE : LINUX, MACOS X

- Principe

EXAMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:

EXAMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:
 1. Tâches **temps réel**: préemptif selon priorité, FIFO ou RR par priorité

EXAMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:
 1. Tâches **temps réel**: préemptif selon priorité, FIFO ou RR par priorité
 2. Autres tâches: temps partagé équitable

EXAMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:
 1. Tâches **temps réel**: préemptif selon priorité, FIFO ou RR par priorité
 2. Autres tâches: temps partagé équitable
 - Système de crédits : chaque processus dispose d'un **crédit** = sa priorité

EXEMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:
 1. Tâches **temps réel**: préemptif selon priorité, FIFO ou RR par priorité
 2. Autres tâches: temps partagé équitable
 - Système de crédits : chaque processus dispose d'un **crédit** = sa priorité
 - 👉 Le processus le plus riche l'emporte (préemptif)

EXEMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:
 1. Tâches **temps réel**: préemptif selon priorité, FIFO ou RR par priorité
 2. Autres tâches: temps partagé équitable
 - Système de crédits : chaque processus dispose d'un **crédit** = sa priorité
 - 👉 Le processus le plus riche l'emporte (préemptif)
 - 👉 Perte de 1 crédit à la fin du quantum

EXEMPLE : LINUX, MACOS X

- **Principe**
 - 2 algorithmes:
 1. Tâches **temps réel**: préemptif selon priorité, FIFO ou RR par priorité
 2. Autres tâches: temps partagé équitable
 - Système de crédits : chaque processus dispose d'un **crédit** = sa priorité
 - 👉 Le processus le plus riche l'emporte (préemptif)
 - 👉 Perte de 1 crédit à la fin du quantum
 - 👉 Si aucun processus **prêt** n'a de crédit, **tous** les processus sont re-crédités
→ $credit' = credit/2 + credit_{init}$

INTER PROCESS COMMUNICATION - IPC

INTER PROCESS COMMUNICATION - IPC

- L'OS garantie l'**indépendance des processus**

INTER PROCESS COMMUNICATION - IPC

- L'OS garantie l'**indépendance des processus**
 - Par l'ordonnanceur CPU

INTER PROCESS COMMUNICATION - IPC

- L'OS garantie l'**indépendance des processus**
 - Par l'ordonnanceur CPU
 - Par la gestion mémoire que l'on verra plus tard

INTER PROCESS COMMUNICATION - IPC

- L'OS garantie l'**indépendance des processus**
 - Par l'ordonnanceur CPU
 - Par la gestion mémoire que l'on verra plus tard
- Un processus peut **communiquer** avec un autre processus ou avec des périphériques (fichiers, imprimantes, réseaux, ...).

INTER PROCESS COMMUNICATION - IPC

- L'OS garantie l'**indépendance des processus**
 - Par l'ordonnanceur CPU
 - Par la gestion mémoire que l'on verra plus tard
- Un processus peut **communiquer** avec un autre processus ou avec des périphériques (fichiers, imprimantes, réseaux, ...).
- Il est alors nécessaire de mettre en oeuvre un mécanisme de **communication interprocessus**.

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
---------	-------------

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
Signal	Un message système est envoyé d'un processus à un autre.

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
Signal	Un message système est envoyé d'un processus à un autre.
Pipe	Un canal unidirectionnel ; les données émises sont accumulées dans un tampon (FIFO).

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
Signal	Un message système est envoyé d'un processus à un autre.
Pipe	Un canal unidirectionnel ; les données émises sont accumulées dans un tampon (FIFO).
File	Lecture/Écriture dans un fichier.

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
Signal	Un message système est envoyé d'un processus à un autre.
Pipe	Un canal unidirectionnel ; les données émises sont accumulées dans un tampon (FIFO).
File	Lecture/Écriture dans un fichier.
Socket	Un flux de données envoyé à travers une interface réseau à un autre processus.

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
Signal	Un message système est envoyé d'un processus à un autre.
Pipe	Un canal unidirectionnel ; les données émises sont accumulées dans un tampon (FIFO).
File	Lecture/Écriture dans un fichier.
Socket	Un flux de données envoyé à travers une interface réseau à un autre processus.
Mémoire Partagée	Espace de mémoire alloué à plusieurs processus.

PRINCIPALES MÉTHODES DE COMMUNICATIONS

Méthode	Description
Signal	Un message système est envoyé d'un processus à un autre.
Pipe	Un canal unidirectionnel ; les données émises sont accumulées dans un tampon (FIFO).
File	Lecture/Écriture dans un fichier.
Socket	Un flux de données envoyé à travers une interface réseau à un autre processus.
Mémoire Partagée	Espace de mémoire alloué à plusieurs processus.
Moniteur/Sémaphore	Une structure de synchronisation pour les processus travaillant sur des ressources partagées.

PROBLÈME DE LA CONCURRENCE

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soit la gestion d'un compte bancaire

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soit la gestion d'un compte bancaire
 - Une variable partagée **balance**

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soit la gestion d'un compte bancaire
 - Une variable partagée `balance`
 - Une fonction `add(1)` (`balance = balance + 1`)

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soit la gestion d'un compte bancaire
 - Une variable partagée `balance`
 - Une fonction `add(1)` (`balance = balance + 1`)
 - Une fonction `sub(1)` (`balance = balance - 1`)

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soit la gestion d'un compte bancaire
 - Une variable partagée `balance`
 - Une fonction `add(1)` (`balance = balance + 1`)
 - Une fonction `sub(1)` (`balance = balance - 1`)
 - Le montant initial du compte est de `9€`

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soient 2 threads en parallèle

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soient 2 threads en parallèle
 - Le premier thread ajoute 10 000 000 fois 1€

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soient 2 threads en parallèle
 - Le premier thread ajoute 10 000 000 fois 1€
 - Le deuxième thread soustrait 10 000 000 fois 1€

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soient 2 threads en parallèle
 - Le premier thread ajoute 10 000 000 fois 1€
 - Le deuxième thread soustrait 10 000 000 fois 1€
- Résultat attendu: balance = 9€

PROBLÈME DE LA CONCURRENCE

```
1 ;balance = balance + 1  
2 mov eax, balance  
3 add eax, 1  
4 mov balance, eax
```

```
1 ;balance = balance - 1  
2 mov eax, balance  
3 sub eax, 1  
4 mov balance, eax
```

- Soient 2 threads en parallèle
 - Le premier thread ajoute 10 000 000 fois 1€
 - Le deuxième thread soustrait 10 000 000 fois 1€
- **Résultat attendu:** balance = 9€
- **Résultat obtenu:** balance = -98599€

COMMENT EXPLIQUER CE CALCUL?

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9
```

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

balance = 9, eax = 9

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

balance = 9, eax = 9

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 10
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9
```

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 10
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 8
```

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 10  
balance = 10, eax = 10
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 8
```

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 10  
balance = 10, eax = 10
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 8  
balance = 8, eax = 8
```

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 10  
balance = 10, eax = 10
```

```
balance = 9, eax = 9  
balance = 9, eax = 8  
balance = 8, eax = 8
```

Après une itération →

COMMENT EXPLIQUER CE CALCUL?

- Les entrelacements se font au niveau du code binaire!
👉 couper *entre chaque instruction assembleur!*

```
1 mov eax, balance  
2 add eax, 1  
3 mov balance, eax
```

```
1 mov eax, balance  
2 sub eax, 1  
3 mov balance, eax
```

```
balance = 9, eax = 9  
balance = 9, eax = 10  
balance = 10, eax = 10
```

```
balance = 9, eax = 9  
balance = 9, eax = 8  
balance = 8, eax = 8
```

Après une itération → **balance = 8**

PROBLÈME DE LA CONCURRENCE

- **Situation de compétition** : erreur dépendant de l'enchaînement temporel d'événements impliquant une ressource partagée

PROBLÈME DE LA CONCURRENCE

- **Situation de compétition** : **erreur** dépendant de l'**enchaînement temporel d'événements** impliquant une ressource partagée
👉 non déterministe

PROBLÈME DE LA CONCURRENCE

- **Situation de compétition** : **erreur** dépendant de l'**enchaînement temporel d'événements** impliquant une ressource partagée
 - 👉 non déterministe
 - 👉 difficile à détecter (**tests**)

PROBLÈME DE LA CONCURRENCE

- **Situation de compétition** : **erreur** dépendant de l'**enchaînement temporel d'événements** impliquant une ressource partagée
 - 👉 non déterministe
 - 👉 difficile à détecter (**tests**)
 - 👉 difficile à corriger (**debug**)

SECTION CRITIQUE

- Lorsqu'il y a des **variables partagées**, il existe des **portions de code** qu'on ne veut pas pouvoir interrompre.

SECTION CRITIQUE

- Lorsqu'il y a des **variables partagées**, il existe des **portions de code** qu'on ne veut pas pouvoir interrompre.
👉 ce sont les **zones du code** qui manipulent des **ressources partagées**

SECTION CRITIQUE

- Lorsqu'il y a des **variables partagées**, il existe des **portions de code** qu'on ne veut pas pouvoir interrompre.
 - 👉 ce sont les **zones du code** qui manipulent des **ressources partagées**
 - 👉 ces zones sont appelées **sections critiques**

EXCLUSION MUTUELLE

EXCLUSION MUTUELLE

- Lorsqu'on déclare une **section critique**, on doit garantir qu'**au plus un seul** processus/thread est dans la section critique.

EXCLUSION MUTUELLE

- Lorsqu'on déclare une **section critique**, on doit garantir qu'**au plus un seul** processus/thread est dans la section critique.
👉 besoin de gérer l'**exclusion mutuelle** des **sections critiques**.

EXCLUSION MUTUELLE

- Lorsqu'on déclare une **section critique**, on doit garantir qu'**au plus un seul** processus/thread est dans la section critique.
 - 👉 besoin de gérer l'**exclusion mutuelle** des **sections critiques**.
 - 👉 **une seule section critique** peut être exécutée à la fois.

EXCLUSION MUTUELLE

- Lorsqu'on déclare une **section critique**, on doit garantir qu'**au plus un seul** processus/thread est dans la section critique.
 - 👉 besoin de gérer l'**exclusion mutuelle** des **sections critiques**.
 - 👉 **une seule section critique** peut être exécutée à la fois.
- Pour résoudre ce problème il faut un **système de verrou**.

SYSTÈMES DE VERROU

SYSTÈMES DE VERROU

1. Processeur non-parallélisé (mono-Processeur, mono-Thread)

SYSTÈMES DE VERROU

1. **Processeur non-parallélisé** (mono-Processeur, mono-Thread)
 - **masquer les interruptions** pendant la section critique

SYSTÈMES DE VERROU

1. **Processeur non-parallélisé** (mono-Processeur, mono-Thread)
 - masquer les interruptions pendant la section critique
 - empêche la préemption

SYSTÈMES DE VERROU

1. **Processeur non-parallélisé** (mono-Processeur, mono-Thread)
 - masquer les interruptions pendant la section critique
 - empêche la préemption
2. **Processeur parallélisé**

SYSTÈMES DE VERROU

1. **Processeur non-parallélisé** (mono-Processeur, mono-Thread)
 - masquer les interruptions pendant la section critique
 - empêche la préemption
2. **Processeur parallélisé**
 - coopération Hardware/Logiciel (quelques instructions atomiques)

SYSTÈMES DE VERROU

1. **Processeur non-parallélisé** (mono-Processeur, mono-Thread)
 - masquer les interruptions pendant la section critique
 - empêche la préemption
2. **Processeur parallélisé**
 - coopération Hardware/Logiciel (quelques instructions atomiques)
 - des mécanismes de haut niveau de l'OS (moniteurs, sémaphores, ...)

LES MONITEURS

LES MONITEURS

- **Un moniteur** est un module constitué de:

LES MONITEURS

- **Un moniteur** est un module constitué de:
 - variables **inaccessibles** de l'extérieur

LES MONITEURS

- **Un moniteur** est un module constitué de:
 - variables **inaccessibles** de l'extérieur
 - fonctions exécutées en **exclusion mutuelle**

LES MONITEURS

- **Un moniteur** est un module constitué de:
 - variables **inaccessibles** de l'extérieur
 - fonctions exécutées en **exclusion mutuelle**
- Dans la **machine virtuelle Java**, on peut définir que certaines **méthodes** d'une même classe sont en **exclusion mutuelle**

LES MONITEURS

- **Un moniteur** est un module constitué de:
 - variables **inaccessibles** de l'extérieur
 - fonctions exécutées en **exclusion mutuelle**
- Dans la **machine virtuelle Java**, on peut définir que certaines **méthodes** d'une même classe sont en **exclusion mutuelle**
 - mot clé **synchronized** → utilisation d'un verrou

LES MONITEURS

- **Un moniteur** est un module constitué de:
 - variables **inaccessibles** de l'extérieur
 - fonctions exécutées en **exclusion mutuelle**
- Dans la **machine virtuelle Java**, on peut définir que certaines **méthodes** d'une même classe sont en **exclusion mutuelle**
 - mot clé **synchronized** → utilisation d'un verrou
 - utilisé pour de **très courtes sections critiques**

LES MONITEURS

```
1 public class Account {  
2     private int balance;  
3  
4     public Account(int balance) {  
5         this.balance = balance;  
6     }  
7  
8     synchronized public void add(int v){  
9         this.balance = this.balance + v;  
10    }  
11  
12    synchronized public void sub(int v){  
13        this.balance = this.balance - v;  
14    }  
15  
16    synchronized public int getBalance() {  
17        return balance;  
18    }  
19 }
```

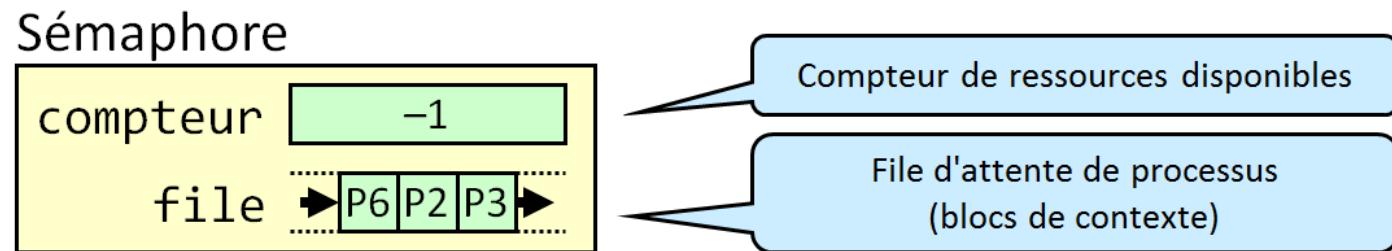
LES MONITEURS

```
1 public class Account {  
2     private int balance;  
3  
4     public Account(int balance) {  
5         this.balance = balance;  
6     }  
7  
8     synchronized public void add(int v){  
9         this.balance = this.balance + v;  
10    }  
11  
12    synchronized public void sub(int v){  
13        this.balance = this.balance - v;  
14    }  
15  
16    synchronized public int getBalance() {  
17        return balance;  
18    }  
19 }
```

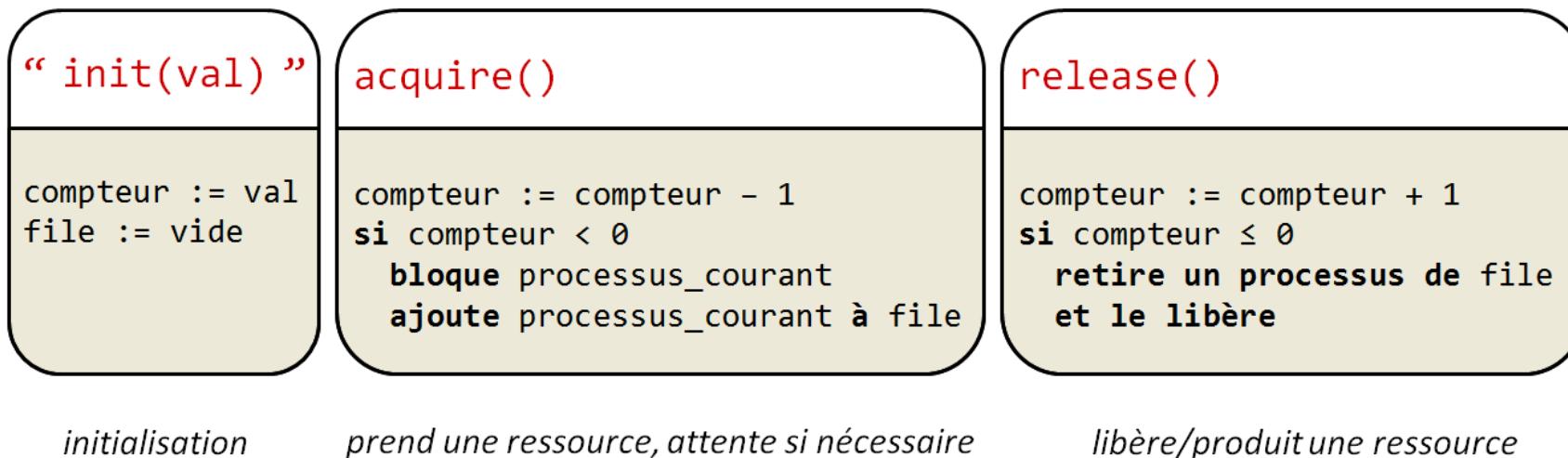
LES MONITEURS

```
1 public class Account {  
2     private int balance;  
3  
4     public Account(int balance) {  
5         this.balance = balance;  
6     }  
7  
8     synchronized public void add(int v){  
9         this.balance = this.balance + v;  
10    }  
11  
12    synchronized public void sub(int v){  
13        this.balance = this.balance - v;  
14    }  
15  
16    synchronized public int getBalance() {  
17        return balance;  
18    }  
19 }
```

LES SÉMAPHORES



3 primitives (opérations **non interruptibles = atomiques**) :



LES SÉMAPHORES

```
1 public class Account {  
2     private int balance;  
3     private Semaphore mutex = new Semaphore(1);  
4  
5     public Account(int balance) {  
6         this.balance = balance;  
7     }  
8  
9     public void add(int v){  
10        this.mutex.acquire();  
11        this.balance = this.balance + v;  
12        this.mutex.release();  
13    }  
14  
15    public void sub(int v){  
16        this.mutex.acquire();  
17        this.balance = this.balance - v;  
18        this.mutex.release();  
19    }  
20 }
```

LES SÉMAPHORES

```
1 public class Account {  
2     private int balance;  
3     private Semaphore mutex = new Semaphore(1);  
4  
5     public Account(int balance) {  
6         this.balance = balance;  
7     }  
8  
9     public void add(int v){  
10        this.mutex.acquire();  
11        this.balance = this.balance + v;  
12        this.mutex.release();  
13    }  
14  
15    public void sub(int v){  
16        this.mutex.acquire();  
17        this.balance = this.balance - v;  
18        this.mutex.release();  
19    }  
20 }
```

LES SÉMAPHORES

```
1 public class Account {  
2     private int balance;  
3     private Semaphore mutex = new Semaphore(1);  
4  
5     public Account(int balance) {  
6         this.balance = balance;  
7     }  
8  
9     public void add(int v){  
10        this.mutex.acquire();  
11        this.balance = this.balance + v;  
12        this.mutex.release();  
13    }  
14  
15    public void sub(int v){  
16        this.mutex.acquire();  
17        this.balance = this.balance - v;  
18        this.mutex.release();  
19    }  
20 }
```

LES SÉMAPHORES

```
1 public class Account {  
2     private int balance;  
3     private Semaphore mutex = new Semaphore(1);  
4  
5     public Account(int balance) {  
6         this.balance = balance;  
7     }  
8  
9     public void add(int v){  
10        this.mutex.acquire();  
11        this.balance = this.balance + v;  
12        this.mutex.release();  
13    }  
14  
15    public void sub(int v){  
16        this.mutex.acquire();  
17        this.balance = this.balance - v;  
18        this.mutex.release();  
19    }  
20 }
```

GESTION DE LA MÉMOIRE

⌚ Fonctionnalités principales d'un OS

ESPACE DE STOCKAGE

ESPACE DE STOCKAGE

- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:

ESPACE DE STOCKAGE

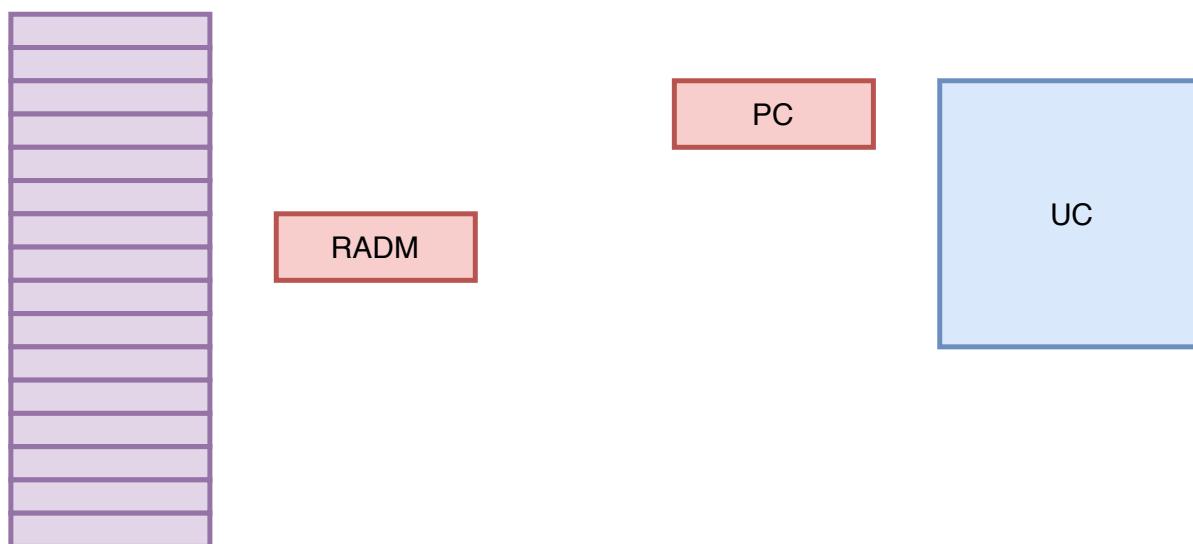
- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur

ESPACE DE STOCKAGE

- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur

ESPACE DE STOCKAGE

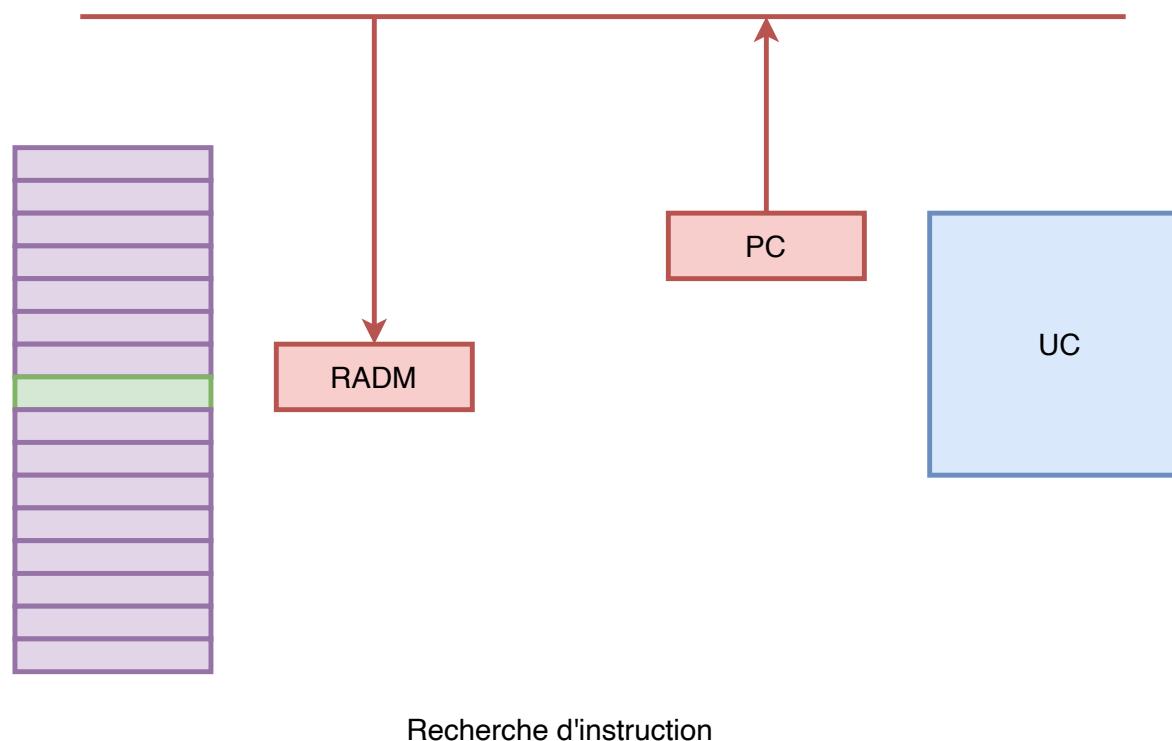
- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur
-



Architecture de Von Neumann

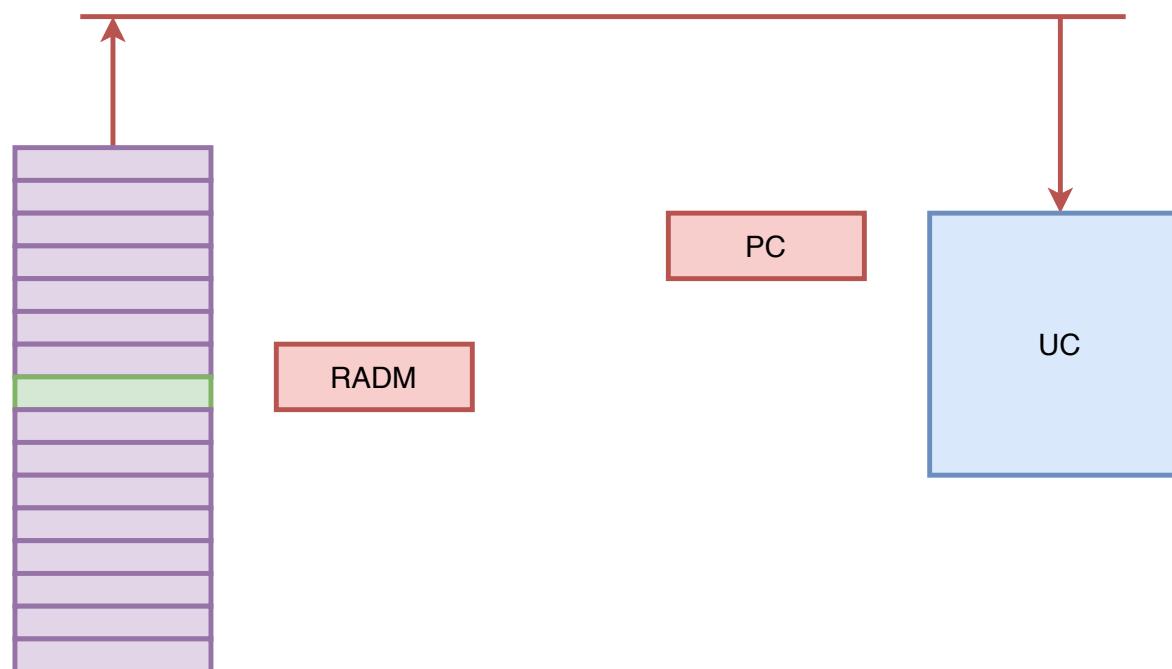
ESPACE DE STOCKAGE

- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur



ESPACE DE STOCKAGE

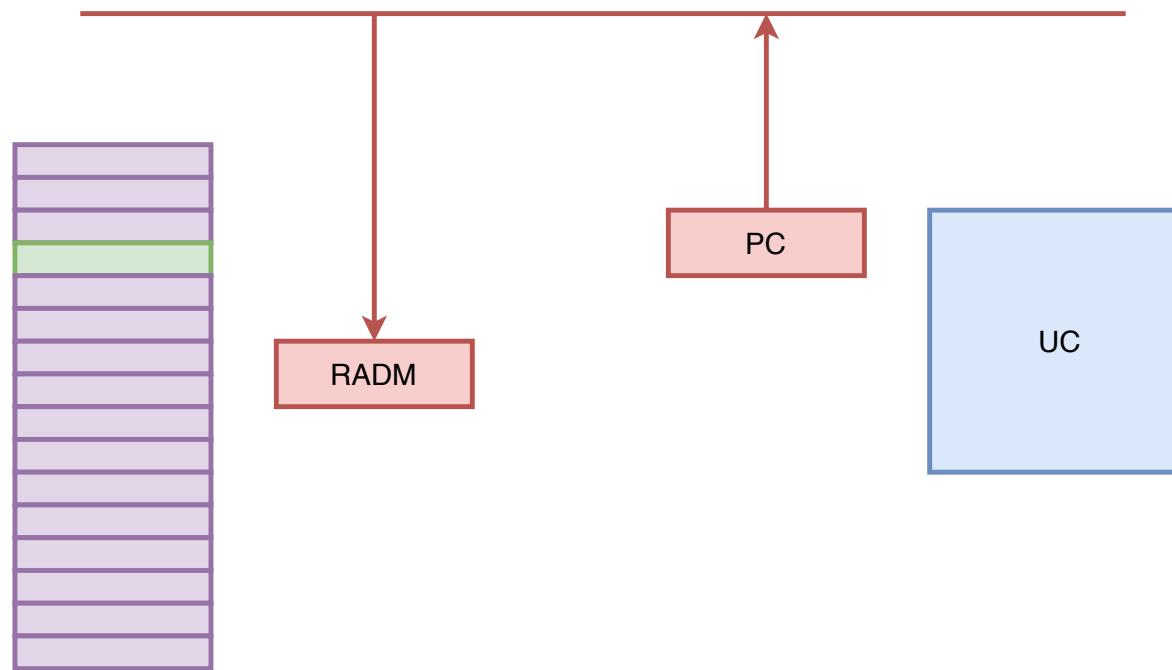
- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur



Recherche d'instruction

ESPACE DE STOCKAGE

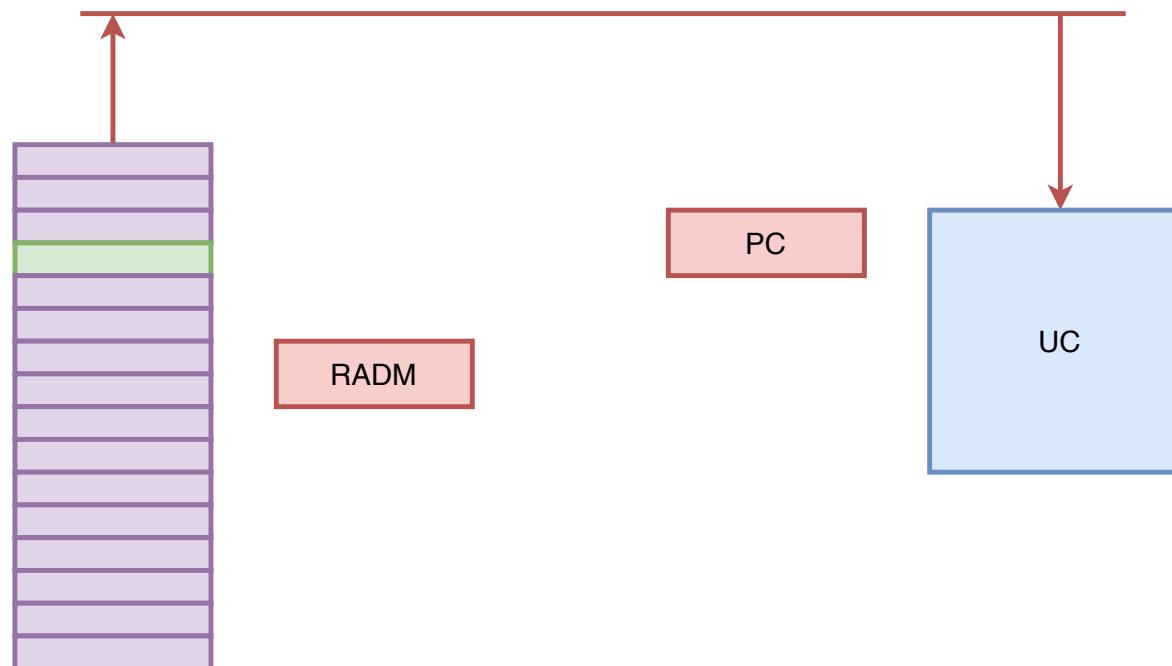
- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur



Lecture d'une donnée

ESPACE DE STOCKAGE

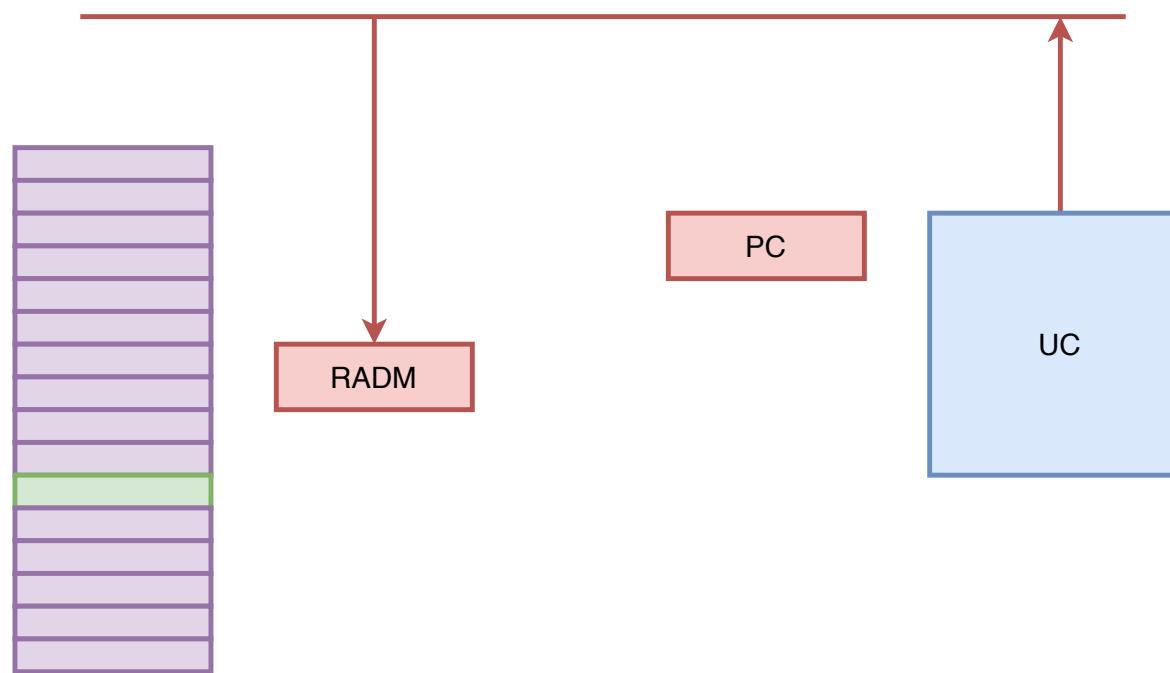
- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur



Lecture d'une donnée

ESPACE DE STOCKAGE

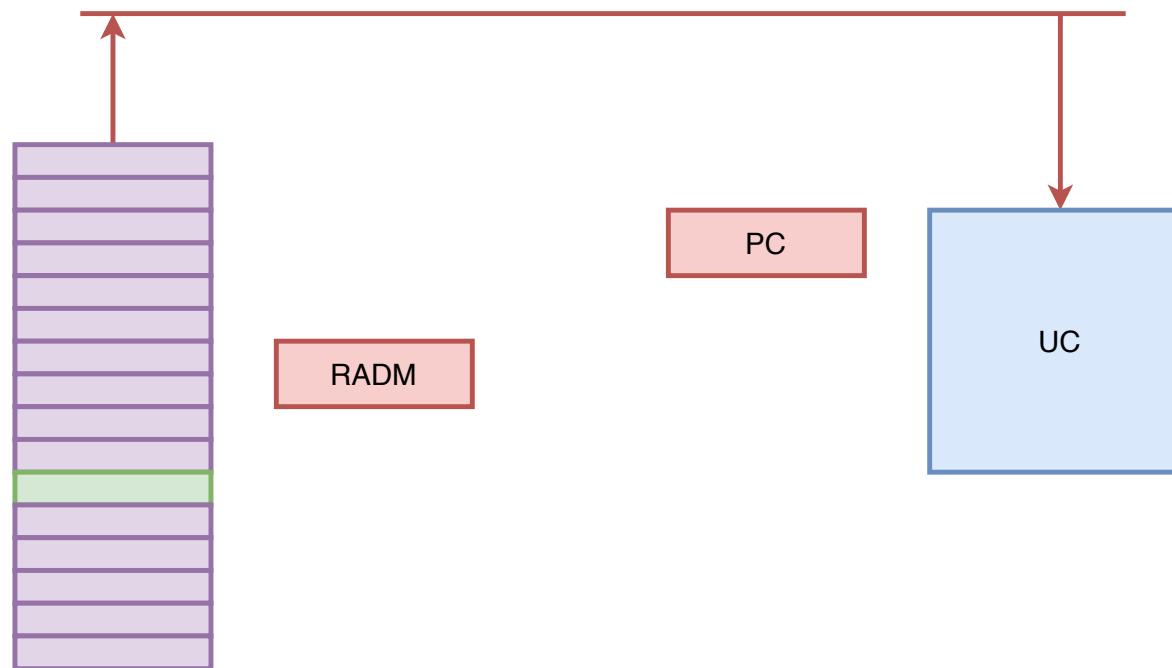
- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur



Lecture d'une donnée

ESPACE DE STOCKAGE

- Ensemble ordonné de **cases** indexée par leur **adresse** (**numéro de la case**) et contenant:
 - Des **instructions** → registre **PC** dans le processeur
 - Des **données** → registre **RADM** dans le processeur



Lecture d'une donnée

LA MÉMOIRE POUR QUI ET POUR QUOI ?

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour les processus

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- **Pour les processus**
 - A la création d'un processus, l'OS crée un **PCB** et **alloue de la mémoire** pour le processus

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour les processus
 - A la création d'un processus, l'OS crée un **PCB** et alloue de la mémoire pour le processus
 - Pour des raisons de **sécurité**, chaque **processus** doit utiliser *une zone mémoire distincte* (*un espace d'adresses*).

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour les processus
 - A la création d'un processus, l'OS crée un **PCB** et alloue de la mémoire pour le processus
 - Pour des raisons de **sécurité**, chaque **processus** doit utiliser *une zone mémoire distincte* (*un espace d'adresses*).
👉 Quel mécanisme d'**allocation** de cet espace ?

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour les processus
 - A la création d'un processus, l'OS crée un **PCB** et alloue de la mémoire pour le processus
 - Pour des raisons de **sécurité**, chaque **processus** doit utiliser *une zone mémoire distincte* (*un espace d'adresses*).
 - 👉 Quel mécanisme d'**allocation** de cet espace ?
 - 👉 Comment assurer la **protection** de cette zone ?

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- **Pour les processus**

- A la création d'un processus, l'OS crée un **PCB** et **alloue de la mémoire pour le processus**
- Pour des raisons de **sécurité**, chaque **processus** doit utiliser *une zone mémoire distincte (un espace d'adresses)*.
 - 👉 Quel mécanisme d'**allocation** de cet espace ?
 - 👉 Comment assurer la **protection** de cette zone ?
 - 👉 Comment assurer la **transparence** de la position de cet espace vis à vis d'un programme ?

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation
 - L'OS aussi a besoin d'espace mémoire

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation
 - L'OS aussi a besoin d'espace mémoire
 - 👉 Code de son Noyau

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation
 - L'OS aussi a besoin d'espace mémoire
 - 👉 Code de son **Noyau**
 - 👉 La table des **interruptions**

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation
 - L'OS aussi a besoin d'espace mémoire
 - 👉 Code de son **Noyau**
 - 👉 La table des **interruptions**
 - 👉 La table des **processus**

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation
 - L'OS aussi a besoin d'espace mémoire
 - 👉 Code de son **Noyau**
 - 👉 La table des **interruptions**
 - 👉 La table des **processus**
 - 👉 Des structures de données (**PCBs** et autres)

LA MÉMOIRE POUR QUI ET POUR QUOI ?

- Pour le système d'exploitation
 - L'OS aussi a besoin d'espace mémoire
 - 👉 Code de son **Noyau**
 - 👉 La table des **interruptions**
 - 👉 La table des **processus**
 - 👉 Des structures de données (**PCBs** et autres)
 - 👉 ...

D'OU VIENNENT LES PROGRAMMES?

D'OU VIENNENT LES PROGRAMMES?

- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire**

D'OU VIENNENT LES PROGRAMMES?

- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire** ... il est placé à un endroit donné dans la mémoire

D'ΟÙ VIENNENT LES PROGRAMMES?

- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire** ... il est placé à un endroit donné dans la mémoire
👉 **Problème** : quelles sont les adresses des variables?

D'ΟÙ VIENNENT LES PROGRAMMES?

- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire** ... il est placé à un endroit donné dans la mémoire
👉 **Problème** : quelles sont les adresses des variables?
- L'**adresse virtuelle** (**adresse logique**) est une adresse utilisée à l'intérieur d'un programme.

D'OU VIENNENT LES PROGRAMMES?

- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire** ... il est placé à un endroit donné dans la mémoire
👉 **Problème** : quelles sont les adresses des variables?
- L'**adresse virtuelle** (**adresse logique**) est une adresse utilisée à l'intérieur d'un programme.
- L'**adresse physique** est utilisée par des puces de **RAM** lors des opérations d'écriture et de lecture.

D'ΟÙ VIENNENT LES PROGRAMMES?

- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire** ... il est placé à un endroit donné dans la mémoire
👉 **Problème** : quelles sont les adresses des variables?
- L'**adresse virtuelle** (**adresse logique**) est une adresse utilisée à l'intérieur d'un programme.
- L'**adresse physique** est utilisée par des puces de **RAM** lors des opérations d'écriture et de lecture.
- Le **Memory Management Unit (MMU)** fait le lien entre les **adresses virtuelles** et les **adresses physiques**

D'ΟÙ VIENNENT LES PROGRAMMES?

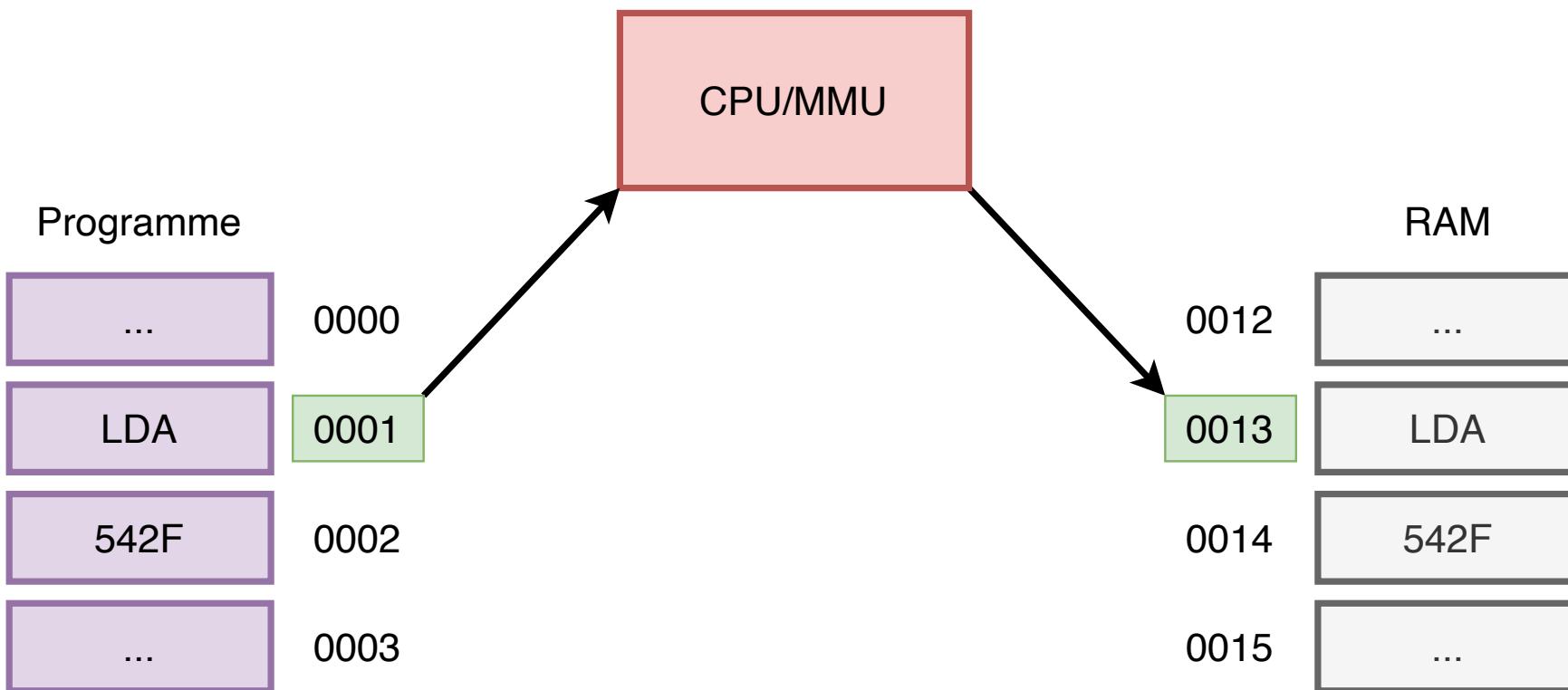
- Un programme (**code + données**) est chargé depuis le **disque** vers la **mémoire** ... il est placé à un endroit donné dans la mémoire
👉 **Problème** : quelles sont les adresses des variables?
- L'**adresse virtuelle** (**adresse logique**) est une adresse utilisée à l'intérieur d'un programme.
- L'**adresse physique** est utilisée par des puces de **RAM** lors des opérations d'écriture et de lecture.
- Le **Memory Management Unit (MMU)** fait le lien entre les **adresses virtuelles** et les **adresses physiques**
 - un composant matériel - souvent dans la **CPU**

MEMORY MANAGEMENT UNIT (MMU)

Le **MMU** a pour rôle de convertir les **adresses logiques** en **adresses physiques**.

MEMORY MANAGEMENT UNIT (MMU)

Le **MMU** a pour rôle de convertir les **adresses logiques** en **adresses physiques**.



INTÉRÊT DES ADRESSES VIRTUELLES

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.
- **Partage efficace** de l'**espace mémoire** entre processus.

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.
- **Partage efficace** de l'**espace mémoire** entre processus.
- Traduction par le **MMU** des **adresses virtuelles** de plusieurs processus en **une même adresse physique** :

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.
- **Partage** efficace de l'**espace mémoire** entre processus.
- Traduction par le **MMU** des **adresses virtuelles** de plusieurs processus en **une même adresse physique** :
 - variables partagées, librairie partagée, ...

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.
- **Partage efficace** de l'**espace mémoire** entre processus.
- Traduction par le **MMU** des **adresses virtuelles** de plusieurs processus en **une même adresse physique** :
 - variables partagées, librairie partagée, ...
- Possibilité de **masquer la réalité** de la **mémoire physique**

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.
- **Partage** efficace de l'**espace mémoire** entre processus.
- Traduction par le **MMU** des **adresses virtuelles** de plusieurs processus en **une même adresse physique** :
 - variables partagées, librairie partagée, ...
- Possibilité de **masquer la réalité** de la **mémoire physique**
 - utilisation du **disque dur** pour étendre la **RAM**.

INTÉRÊT DES ADRESSES VIRTUELLES

- Possibilité d'utiliser des **adresses virtuelles** d'une **taille différente** de celle des **adresses physiques**.
- **Partage efficace de l'espace mémoire** entre processus.
- Traduction par le **MMU** des **adresses virtuelles** de plusieurs processus en **une même adresse physique** :
 - variables partagées, librairie partagée, ...
- Possibilité de **masquer la réalité** de la **mémoire physique**
 - utilisation du **disque dur** pour étendre la **RAM**.
- **Protection des espaces mémoire** des processus.

STRATÉGIES D'ALLOCATION DE MÉMOIRE

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.
- Deux stratégies possibles:

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.
- Deux stratégies possibles:
 1. **Allocation contiguë** de cases mémoire (**par partition**)

STRATÉGIES D'ALLOCATION DE MÉMOIRE

- Il faut choisir une stratégie pour **allouer et libérer la mémoire** en fonction des besoins des processus.
- Deux stratégies possibles:
 1. **Allocation contiguë de cases mémoire (par partition)**
 2. **Allocation non contiguë (par pagination)**

ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire

ALLOCATION PAR PARTITION

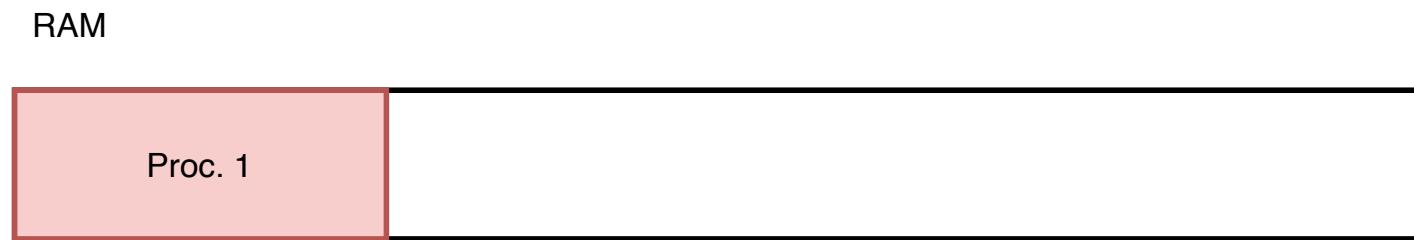
- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire

RAM



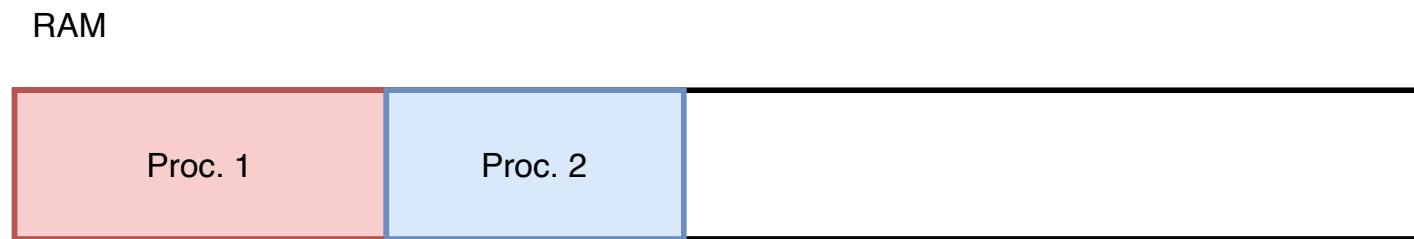
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



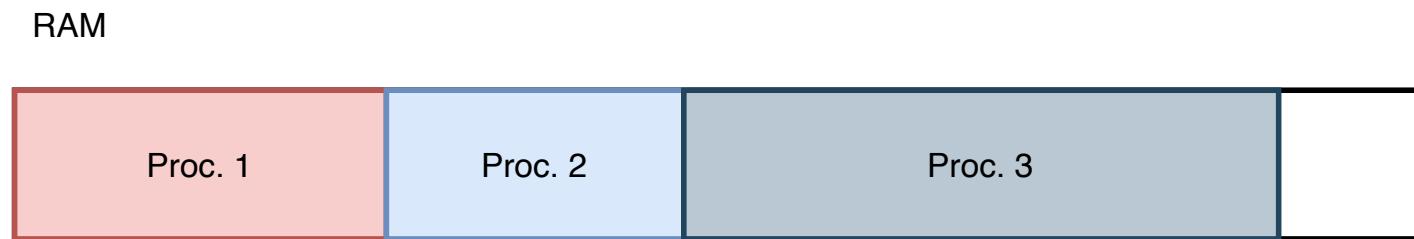
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



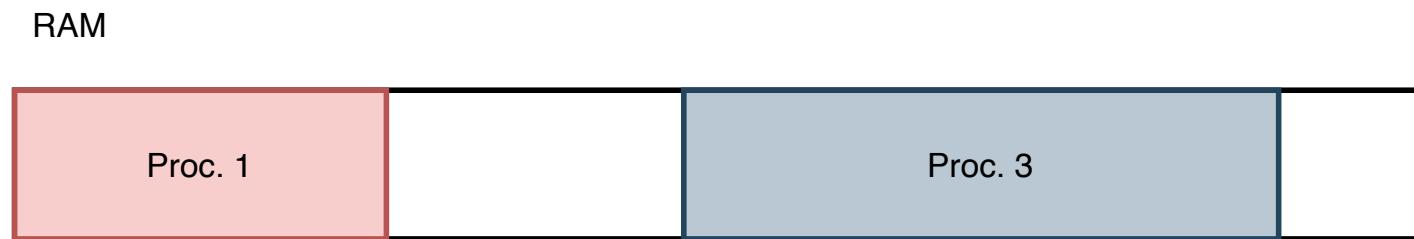
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



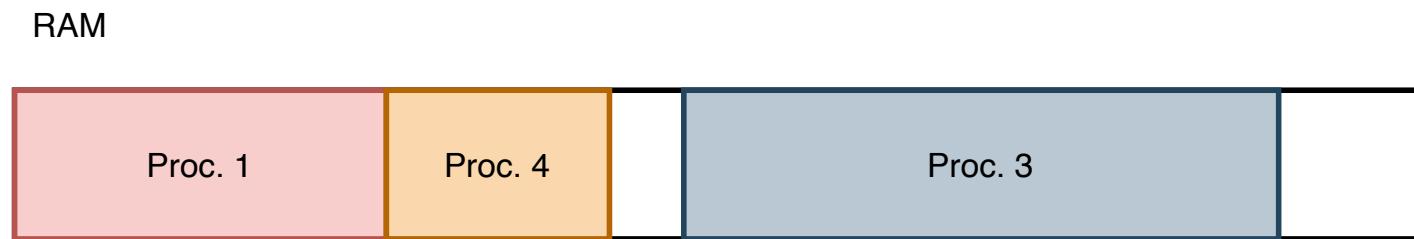
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



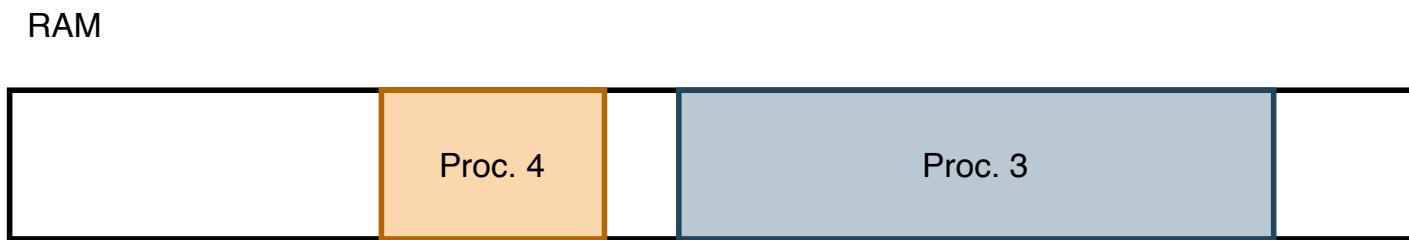
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



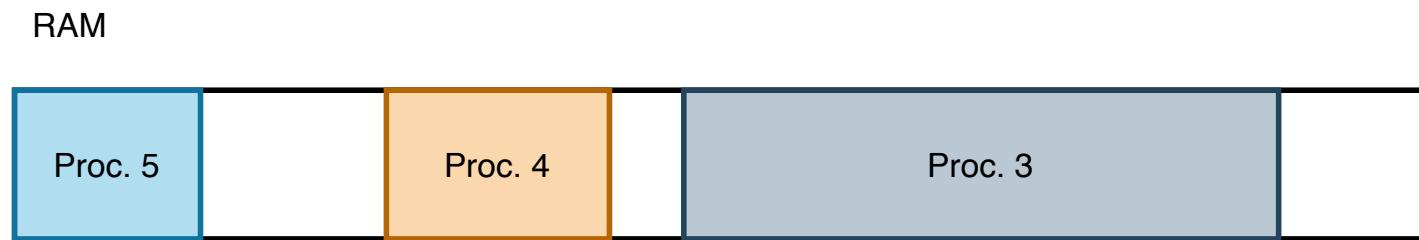
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



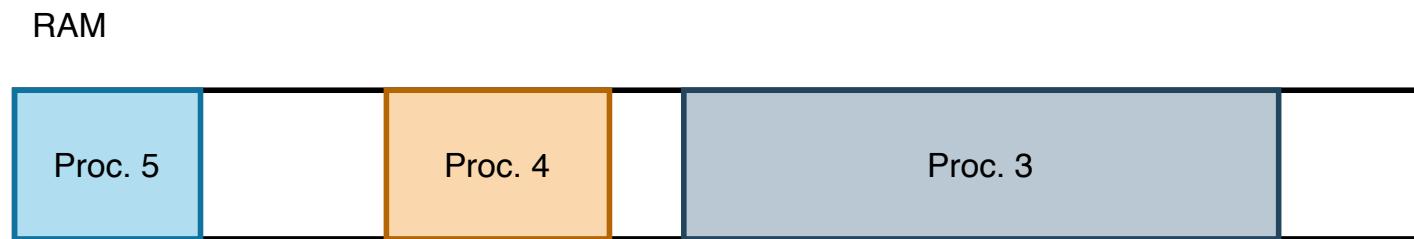
ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



ALLOCATION PAR PARTITION

- **Principe** : Multiprogrammation et temps partagé
👉 placer les processus dans la mémoire



Problèmes :

Des trous apparaissent → **fragmentation**

ALLOCATION PAR PARTITION

Problèmes :

Des trous apparaissent → **fragmentation**

ALLOCATION PAR PARTITION

Problèmes :

Des trous apparaissent → **fragmentation**

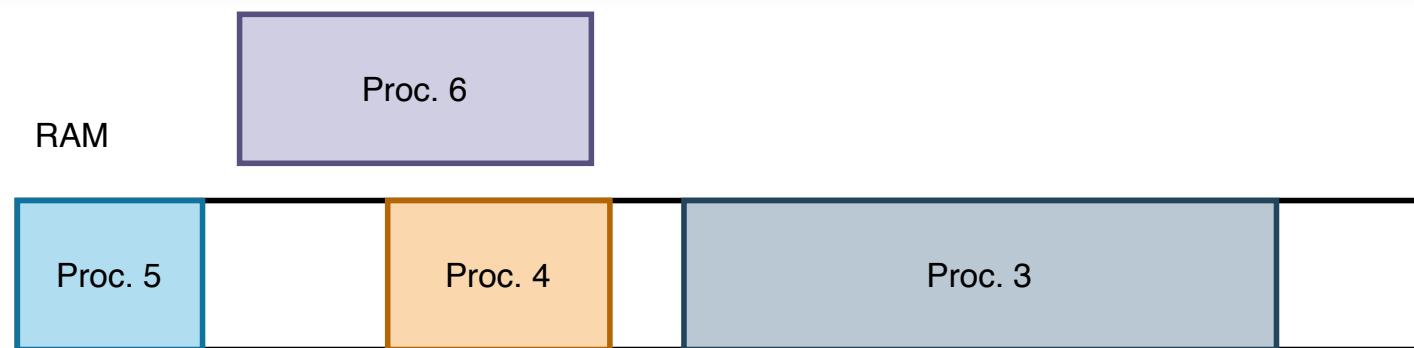
Les processus trop gros ne peuvent pas rentrer → **défragmenter**

ALLOCATION PAR PARTITION

Problèmes :

Des trous apparaissent → **fragmentation**

Les processus trop gros ne peuvent pas rentrer → **défragmenter**

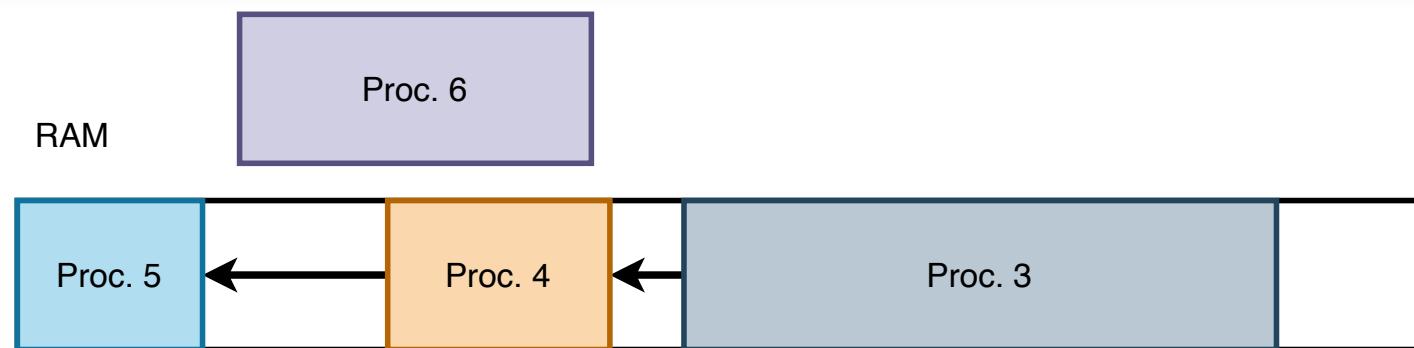


ALLOCATION PAR PARTITION

Problèmes :

Des trous apparaissent → **fragmentation**

Les processus trop gros ne peuvent pas rentrer → **défragmenter**

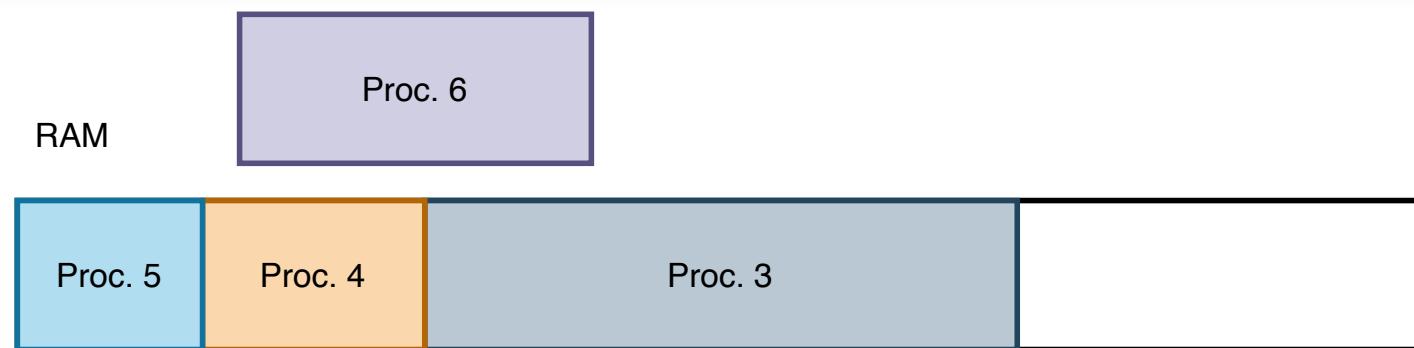


ALLOCATION PAR PARTITION

Problèmes :

Des trous apparaissent → **fragmentation**

Les processus trop gros ne peuvent pas rentrer → **défragmenter**

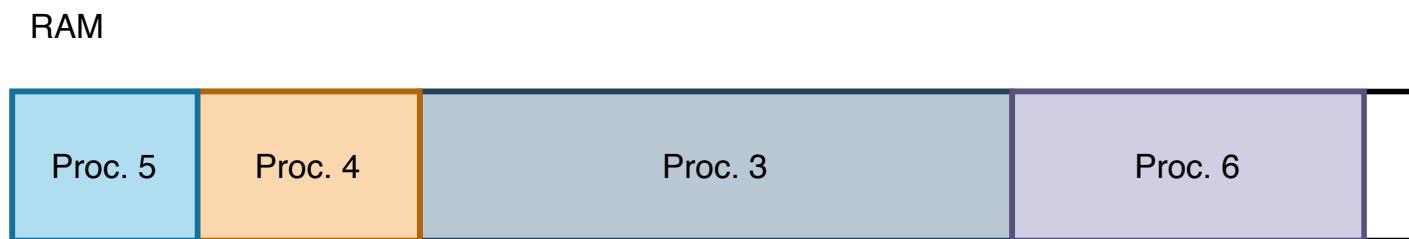


ALLOCATION PAR PARTITION

Problèmes :

Des trous apparaissent → **fragmentation**

Les processus trop gros ne peuvent pas rentrer → **défragmenter**



ALLOCATION PAR PARTITION

- Avantages

ALLOCATION PAR PARTITION

- **Avantages**
 - ✓ Simplicité matérielle

ALLOCATION PAR PARTITION

- **Avantages**
 - ✓ Simplicité matérielle
- **Inconvénients**

ALLOCATION PAR PARTITION

- **Avantages**
 - ✓ Simplicité matérielle
- **Inconvénients**
 - ✗ Fragmentation

ALLOCATION PAR PARTITION

- **Avantages**
 - ✓ Simplicité matérielle
- **Inconvénients**
 - ✗ Fragmentation
 - ✗ Taille fixe des espaces mémoire (taille max à ne pas dépasser)

ALLOCATION PAR PAGINATION

Principe :

ALLOCATION PAR PAGINATION

Principe :

- Découper **la mémoire physique** en **blocs** (**frame**) de taille T_c , appelés **cadres de pages**.

ALLOCATION PAR PAGINATION

Principe :

- Découper la mémoire physique en blocs (frame) de taille T_c , appelés cadres de pages.
- Découper l'espace mémoire utilisé par un processus (espace logique) en paquets de x pages de taille T_c .

ALLOCATION PAR PAGINATION

Principe :

- Découper la mémoire physique en blocs (frame) de taille T_c , appelés cadres de pages.
- Découper l'espace mémoire utilisé par un processus (espace logique) en paquets de x pages de taille T_c .
 - chaque page a la même taille qu'un bloc

ALLOCATION PAR PAGINATION

ALLOCATION PAR PAGINATION

RAM

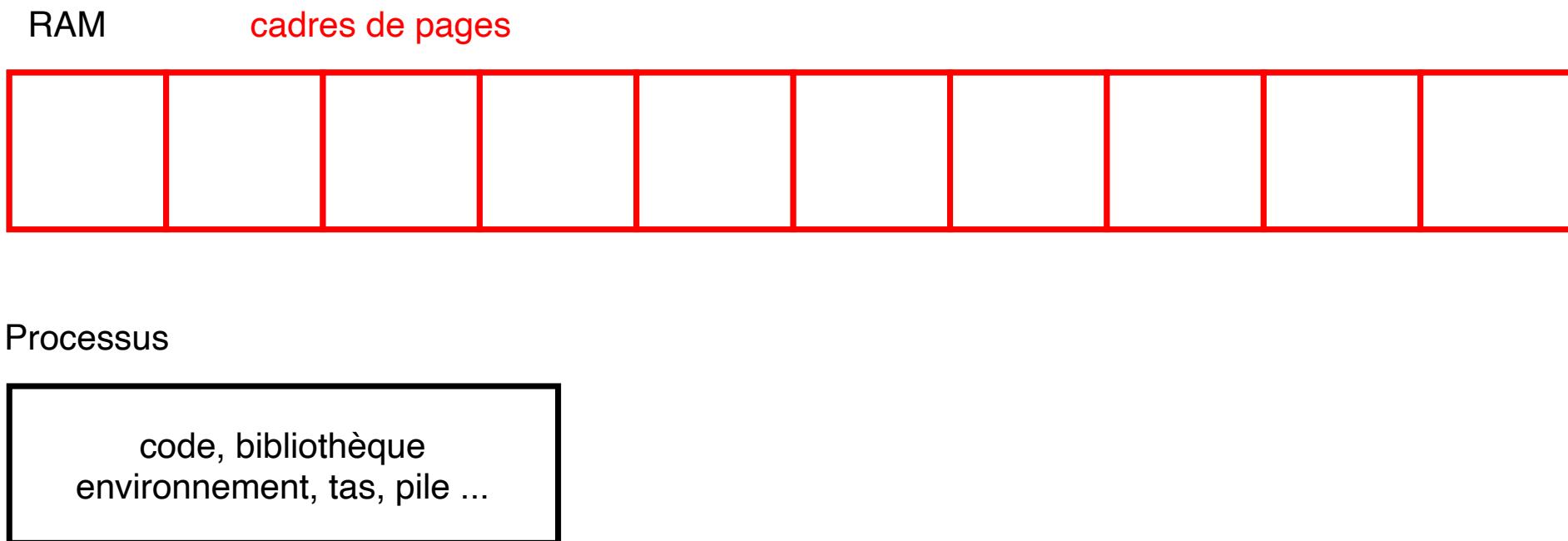


Processus

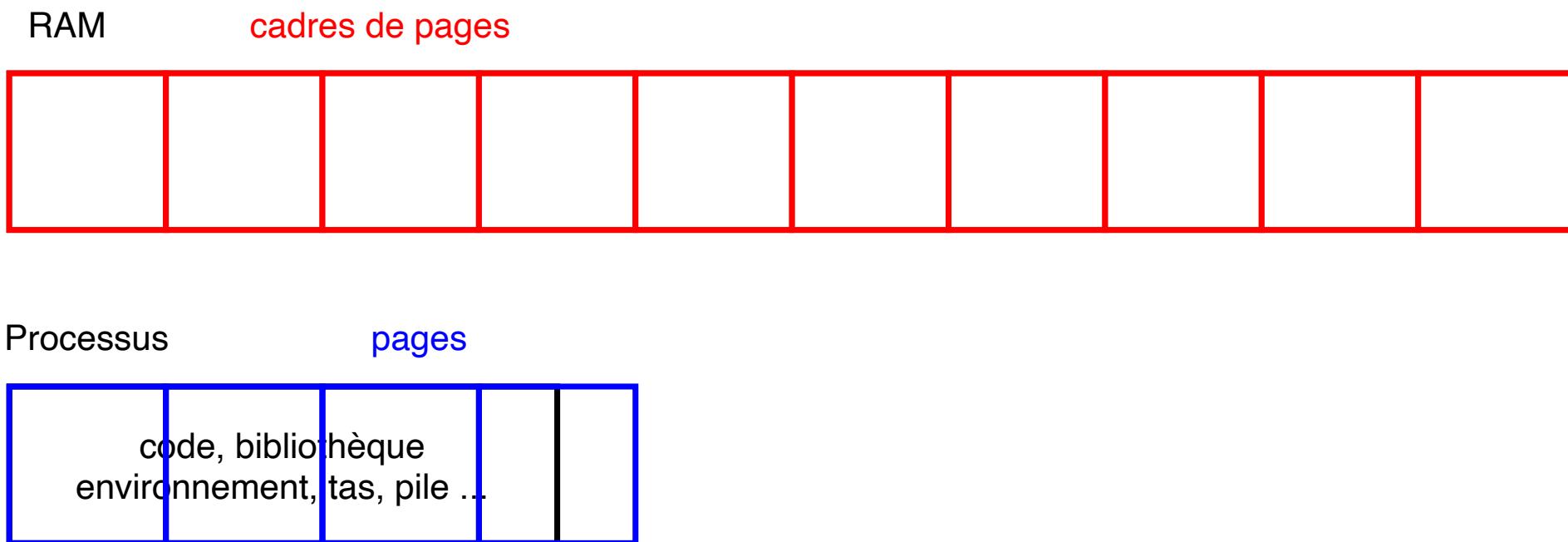
code, bibliothèque
environnement, tas, pile ...

A smaller rectangular box with a black border, containing the text "code, bibliothèque", "environnement, tas, pile ...", which represents the memory footprint of a process.

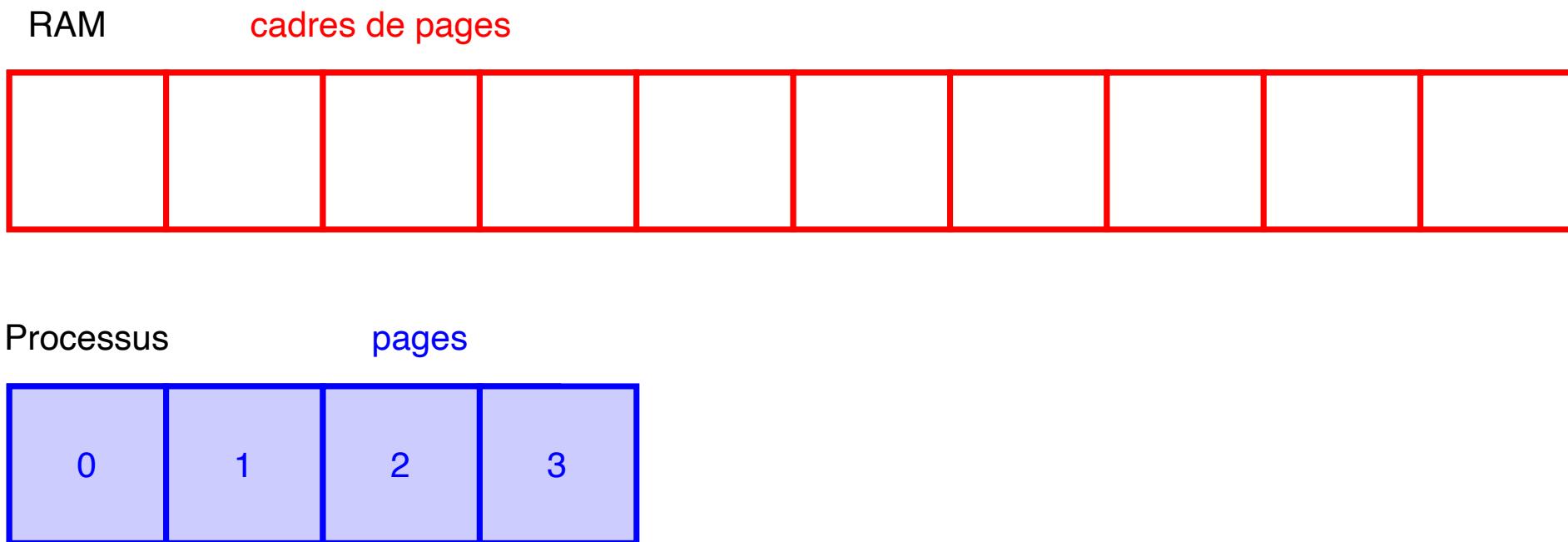
ALLOCATION PAR PAGINATION



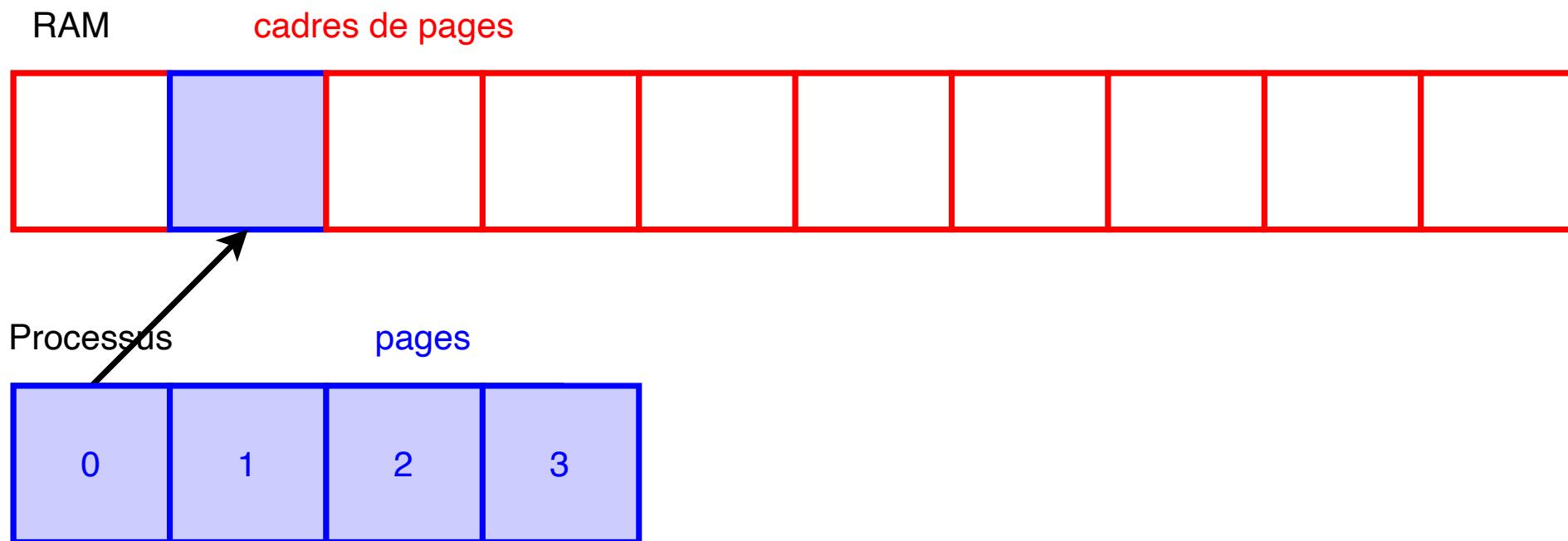
ALLOCATION PAR PAGINATION



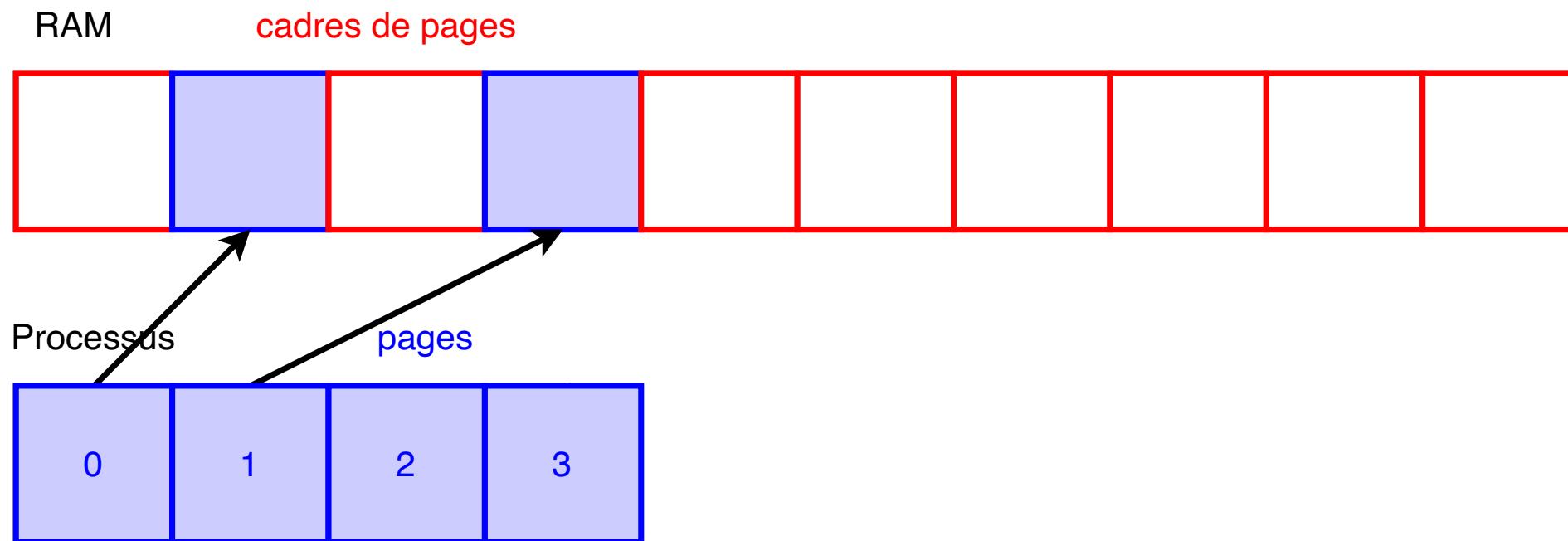
ALLOCATION PAR PAGINATION



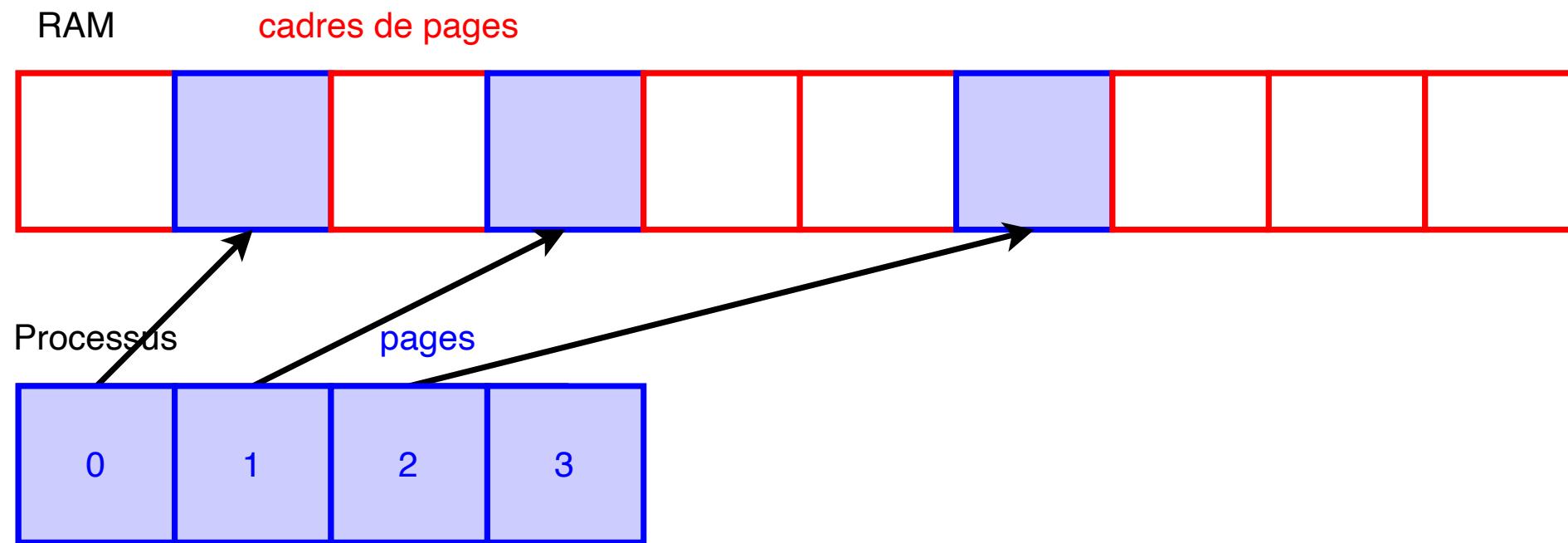
ALLOCATION PAR PAGINATION



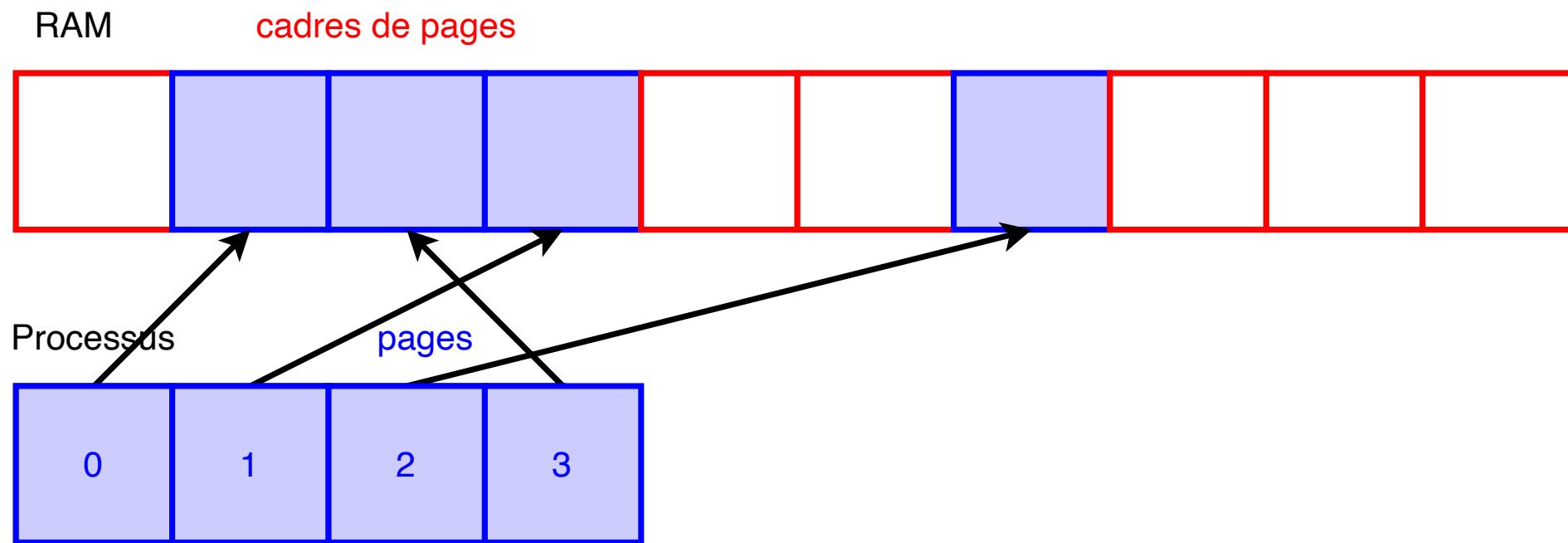
ALLOCATION PAR PAGINATION



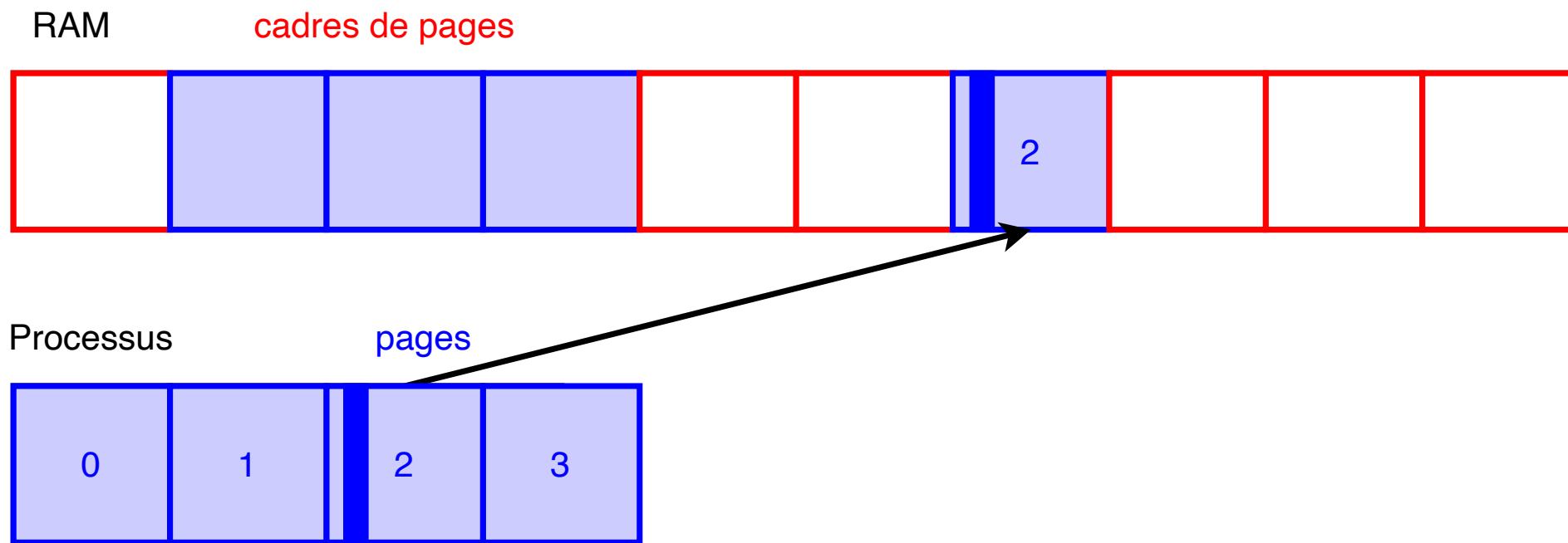
ALLOCATION PAR PAGINATION



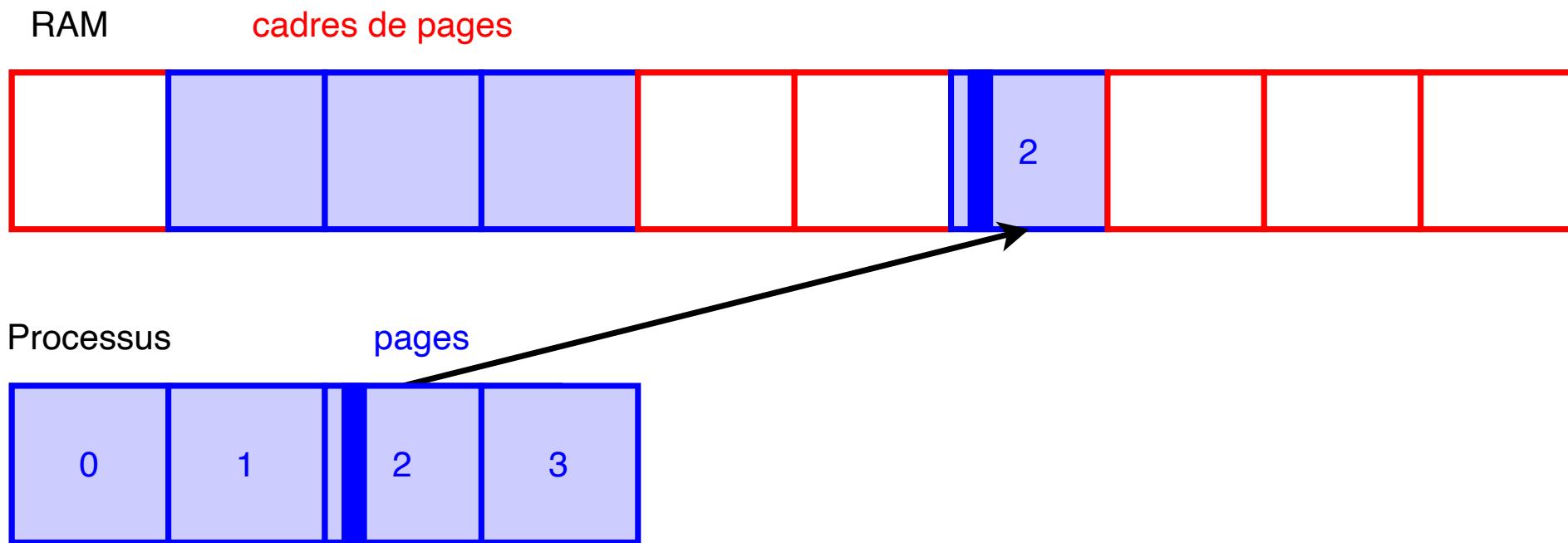
ALLOCATION PAR PAGINATION



ALLOCATION PAR PAGINATION



ALLOCATION PAR PAGINATION



Adressage :

Déterminer l'adresse **physique** à partir de l'adresse **logique**

ALLOCATION PAR PAGINATION

Adressage :

Déterminer l'adresse **physique** à partir de l'adresse **logique**

ALLOCATION PAR PAGINATION

Adressage :

Déterminer l'adresse **physique** à partir de l'adresse **logique**

- **Adresse logique**
 - Numéro de page (n bits) + décalage (m bits)

ALLOCATION PAR PAGINATION

Adressage :

Déterminer l'adresse **physique** à partir de l'adresse **logique**

- **Adresse logique**
 - Numéro de page (n bits) + décalage (m bits)

- **Table des pages**

Chaque processus maintient une liste:

- numéro de page \leftrightarrow numéro de cadre
- c'est la **table des pages**

RÉSOLUTION D'ADRESSE

RÉSOLUTION D'ADRESSE

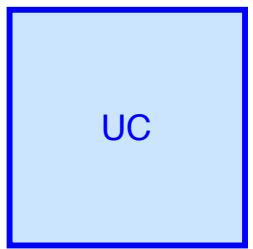
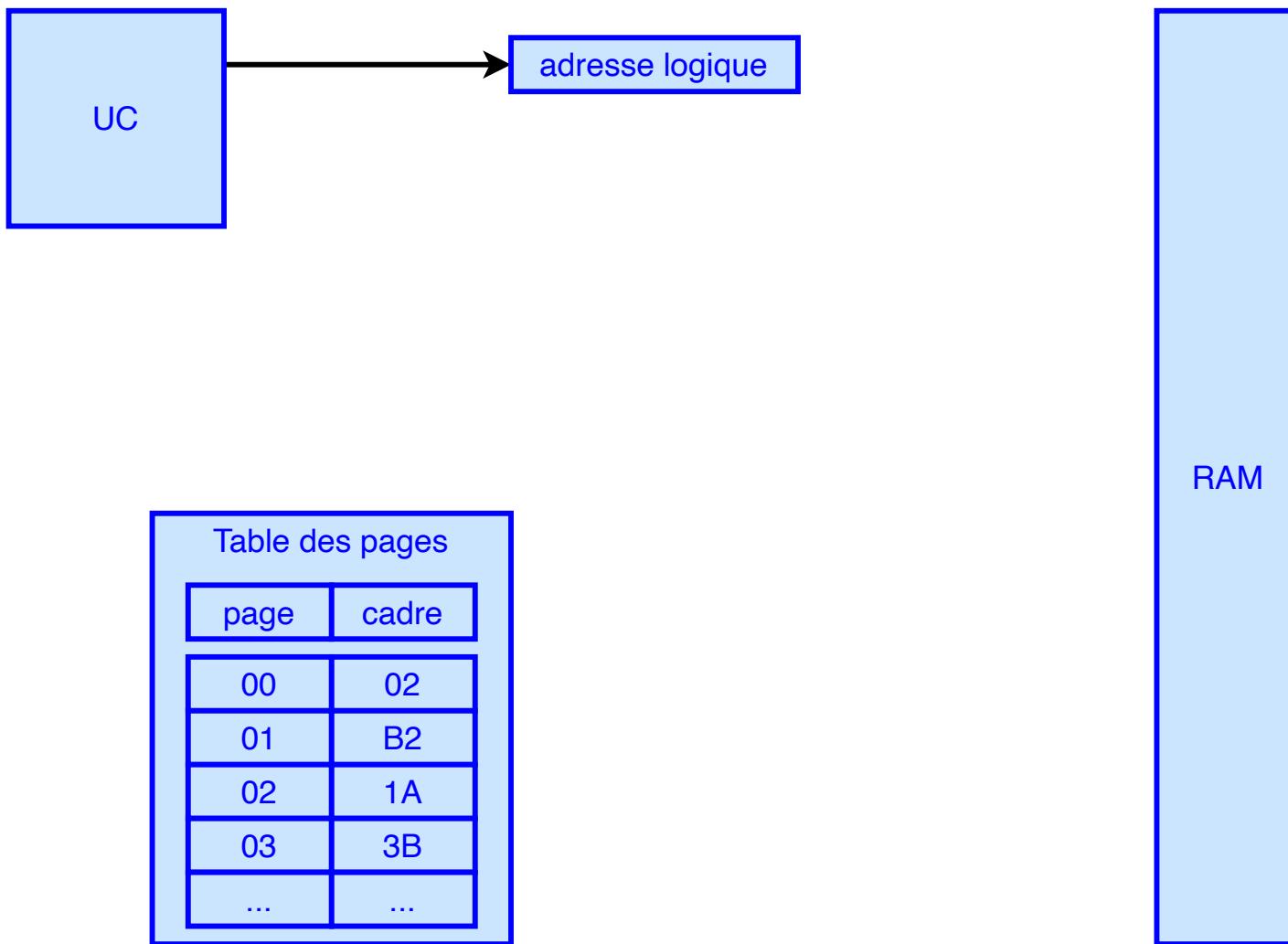
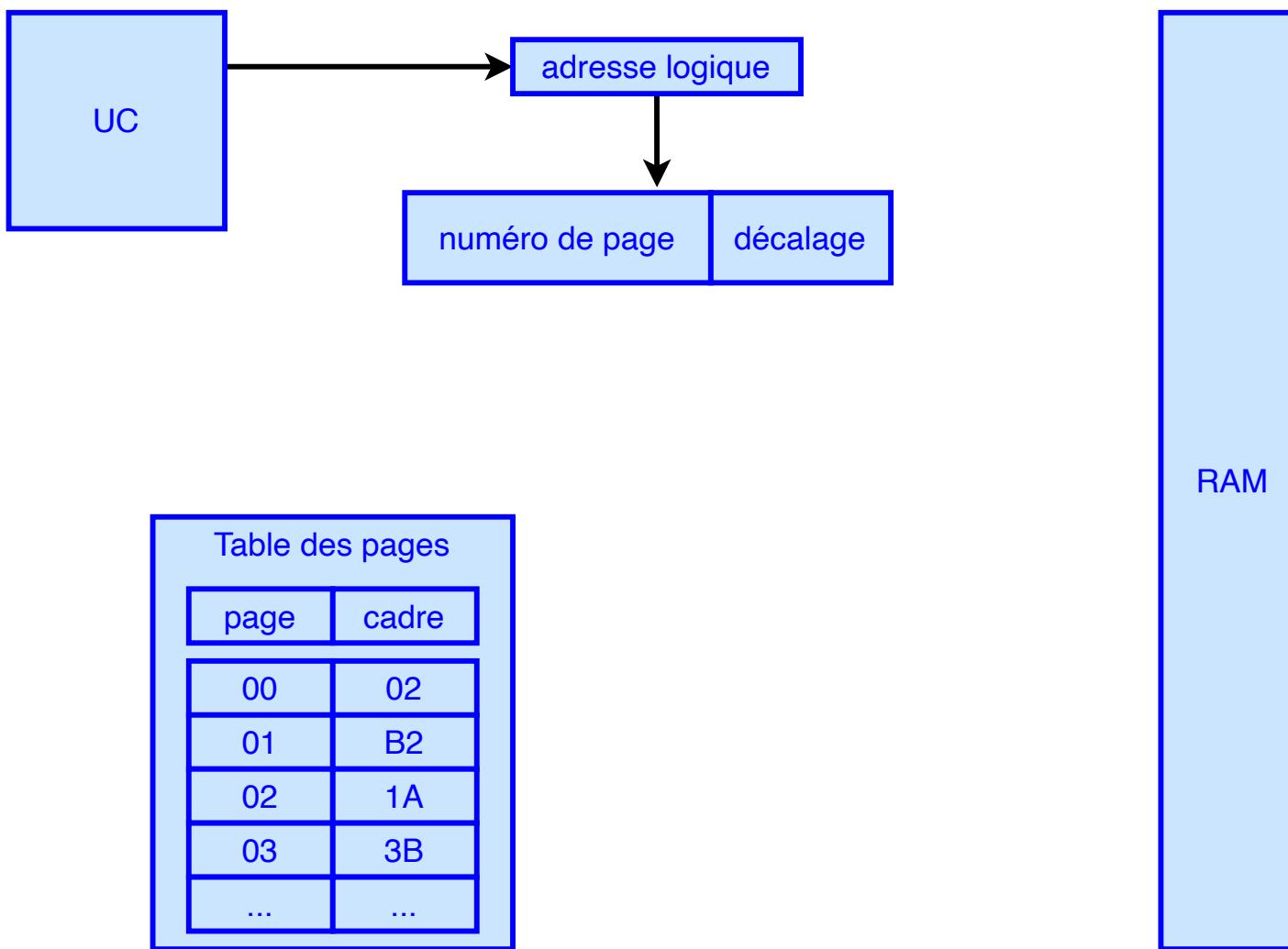


Table des pages	
page	cadre
00	02
01	B2
02	1A
03	3B
...	...

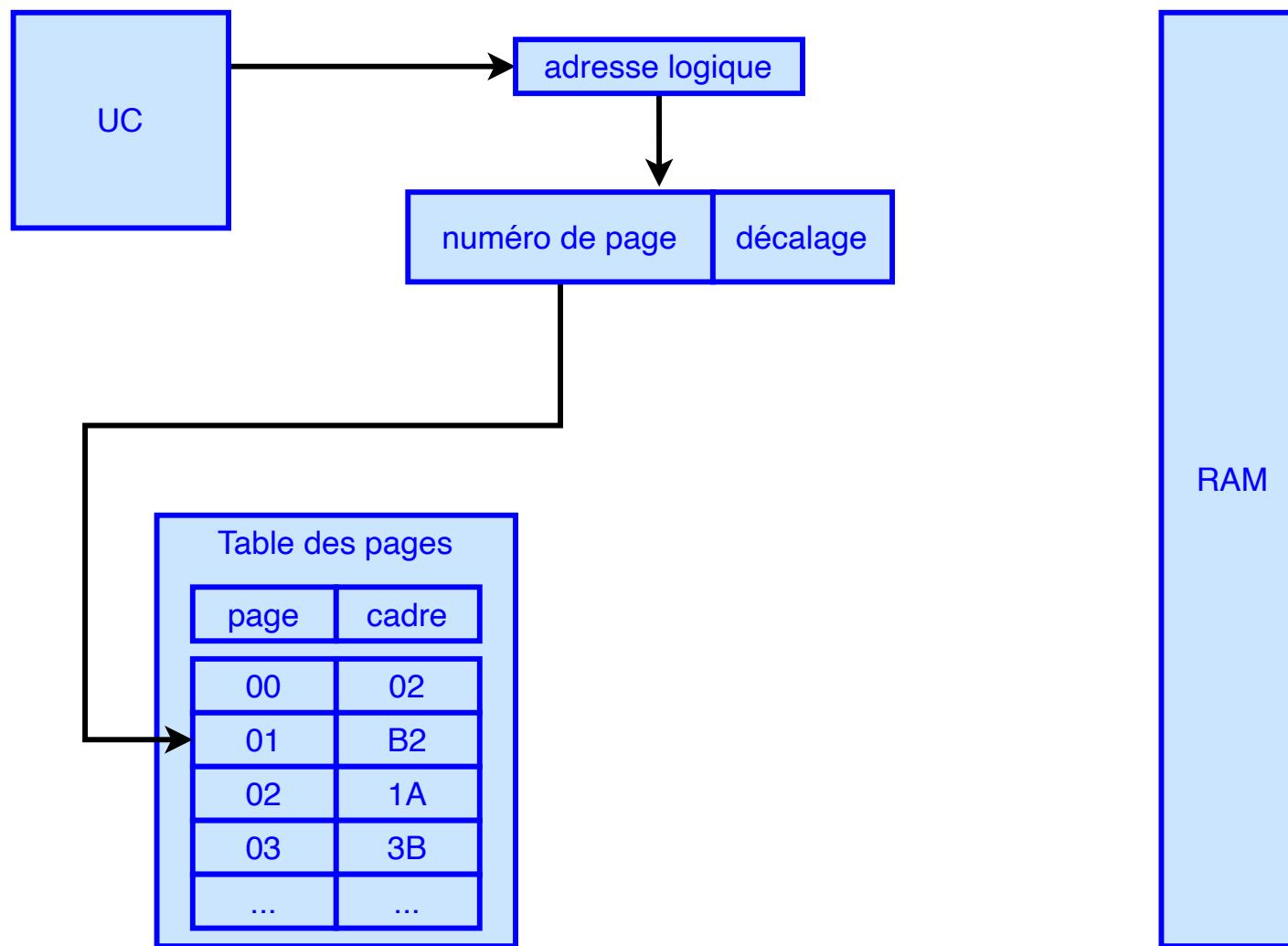
RÉSOLUTION D'ADRESSE



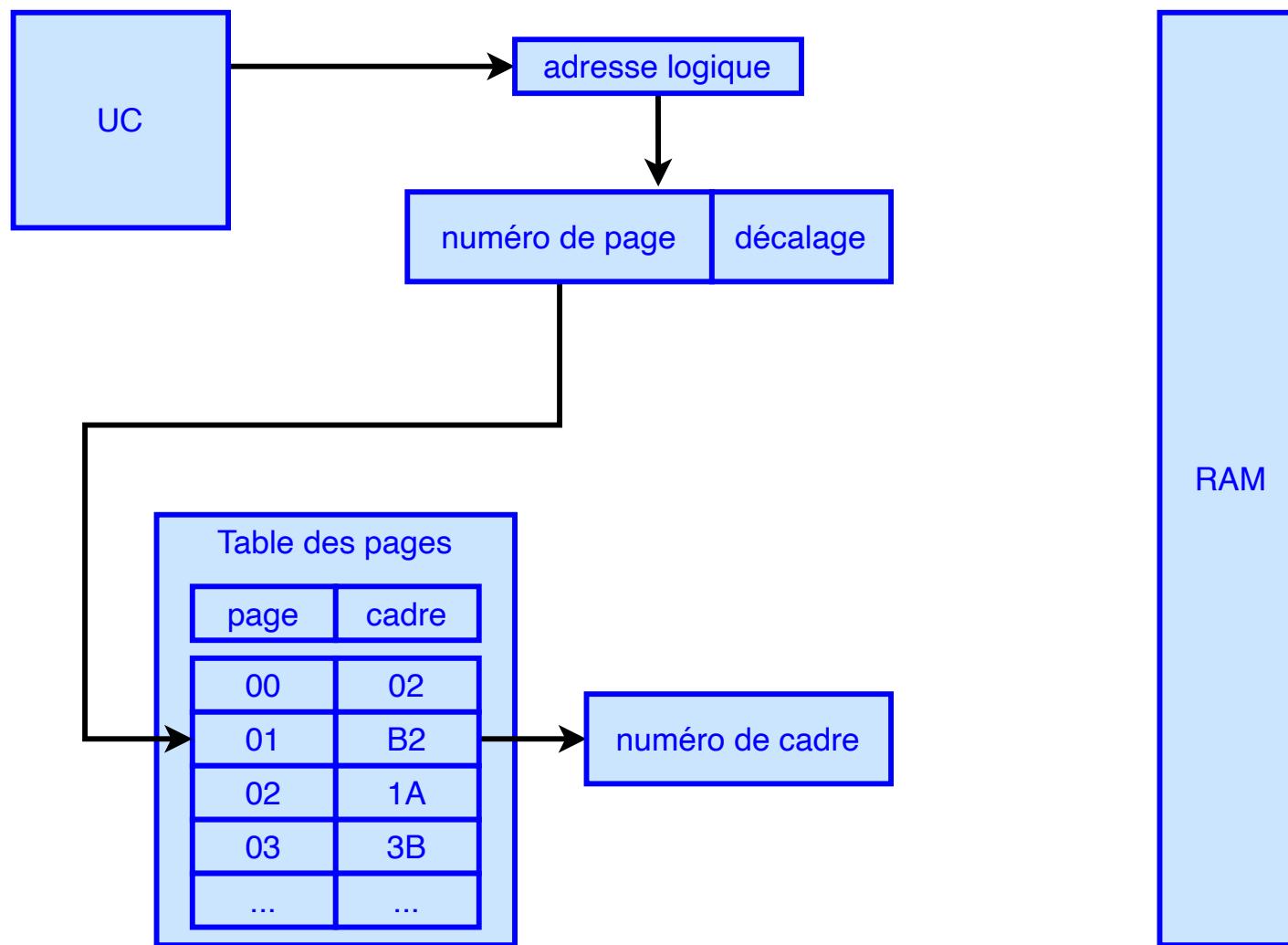
RÉSOLUTION D'ADRESSE



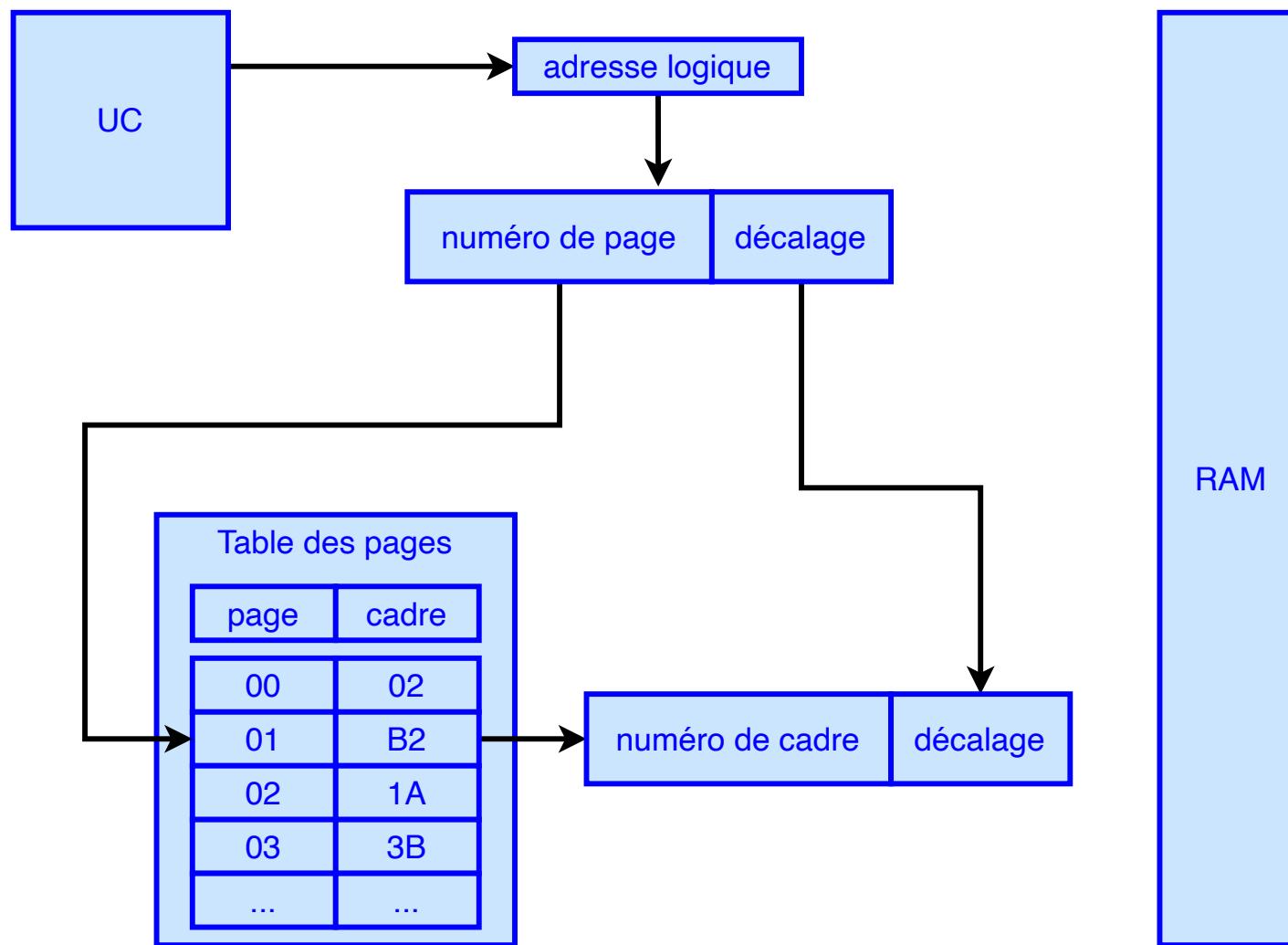
RÉSOLUTION D'ADRESSE



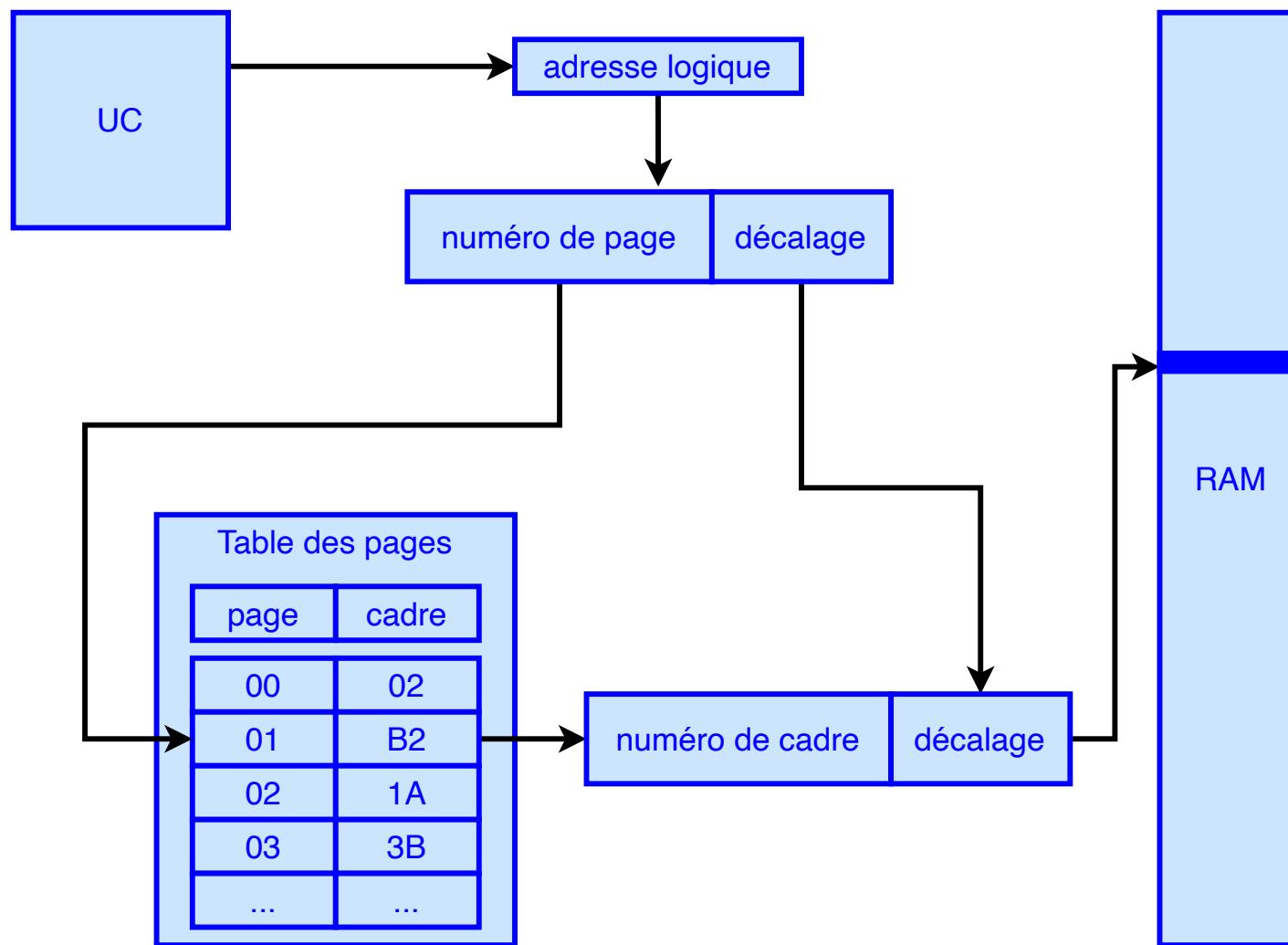
RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE



RÉSOLUTION D'ADRESSE

Memory Management Unit

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la MMU

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la **MMU**
 - L'OS charge **la table des pages du processus** dans la **MMU**

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la **MMU**
 - L'OS charge **la table des pages du processus** dans la **MMU**
 - Pas de calcul d'adresse au niveau de l'OS

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la **MMU**
 - L'OS charge **la table des pages du processus** dans la **MMU**
 - Pas de calcul d'adresse au niveau de l'OS

Allocation des cadres

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la **MMU**
 - L'OS charge **la table des pages du processus** dans la **MMU**
 - Pas de calcul d'adresse au niveau de l'OS

Allocation des cadres

- Ne pas allouer le même cadre à deux processus différents

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la **MMU**
 - L'OS charge **la table des pages du processus** dans la **MMU**
 - Pas de calcul d'adresse au niveau de l'OS

Allocation des cadres

- Ne pas allouer le même cadre à deux processus différents
 - L'OS doit savoir quel processus utilise quel cadre

RÉSOLUTION D'ADRESSE

Memory Management Unit

- En pratique, géré au niveau matériel : la **MMU**
 - L'OS charge **la table des pages du processus** dans la **MMU**
 - Pas de calcul d'adresse au niveau de l'OS

Allocation des cadres

- Ne pas allouer le même cadre à deux processus différents
 - L'OS doit savoir quel processus utilise quel cadre
 - **Table des cadres de pages libres**

PAGINATION HIÉRARCHIQUE

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits = 2,5 Mo par processus

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits = 2,5 Mo par processus
 - Allocation et commutation plus coûteuse en temps

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits = 2,5 Mo par processus
 - Allocation et commutation plus coûteuse en temps
- Pages trop grosses (ex: $n = 10, m = 22$)

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits = 2,5 Mo par processus
 - Allocation et commutation plus coûteuse en temps
- Pages trop grosses (ex: $n = 10, m = 22$)
 - Fragmentation = 2^{m-1} → place mémoire perdue

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits = 2,5 Mo par processus
 - Allocation et commutation plus coûteuse en temps
- Pages trop grosses (ex: $n = 10, m = 22$)
 - Fragmentation = 2^{m-1} → place mémoire perdue
 - Pas gérable au niveau du MMU

PAGINATION HIÉRARCHIQUE

Problème → Grand espace d'adressage (ex: 32 bits)

- Trop de pages (ex: $n = 20, m = 12$)
 - Grande table des pages → place mémoire perdue
 - 2^{20} lignes de 20 bits = 2,5 Mo par processus
 - Allocation et commutation plus coûteuse en temps
- Pages trop grosses (ex: $n = 10, m = 22$)
 - Fragmentation = 2^{m-1} → place mémoire perdue
 - Pas gérable au niveau du MMU
 - 2^{21} octets = 2 Mo par processus

PAGINATION HIÉRARCHIQUE

Pagination à 2 niveaux (ou plus)

PAGINATION HIÉRARCHIQUE

Pagination à 2 niveaux (ou plus)

- Paginer la table des pages (et ne charger que les tables utiles)

PAGINATION HIÉRARCHIQUE

Pagination à 2 niveaux (ou plus)

- Paginer la table des pages (et ne charger que les tables utiles)
 - Réduire la fragmentation due aux pages

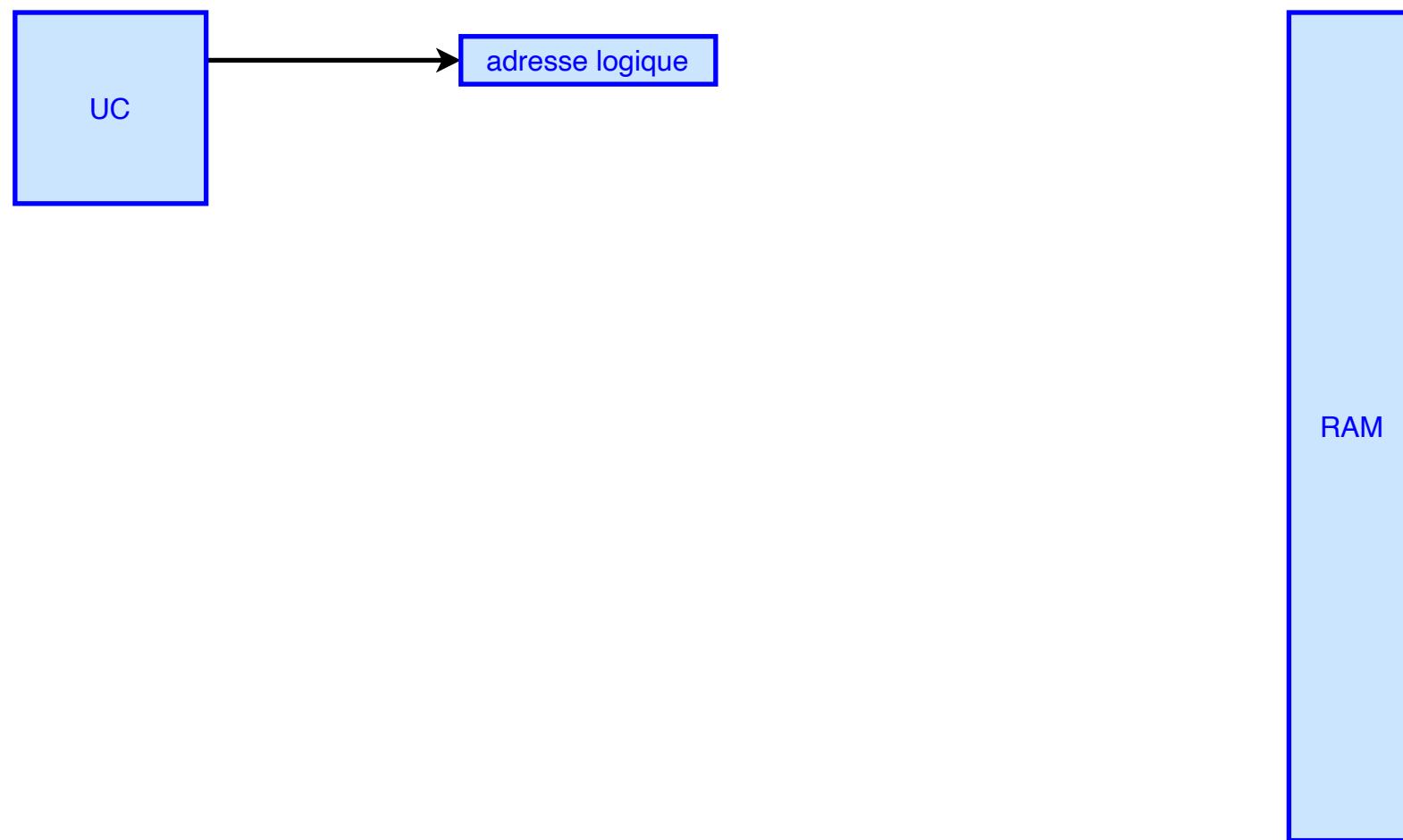
PAGINATION HIÉRARCHIQUE

Pagination à 2 niveaux (ou plus)

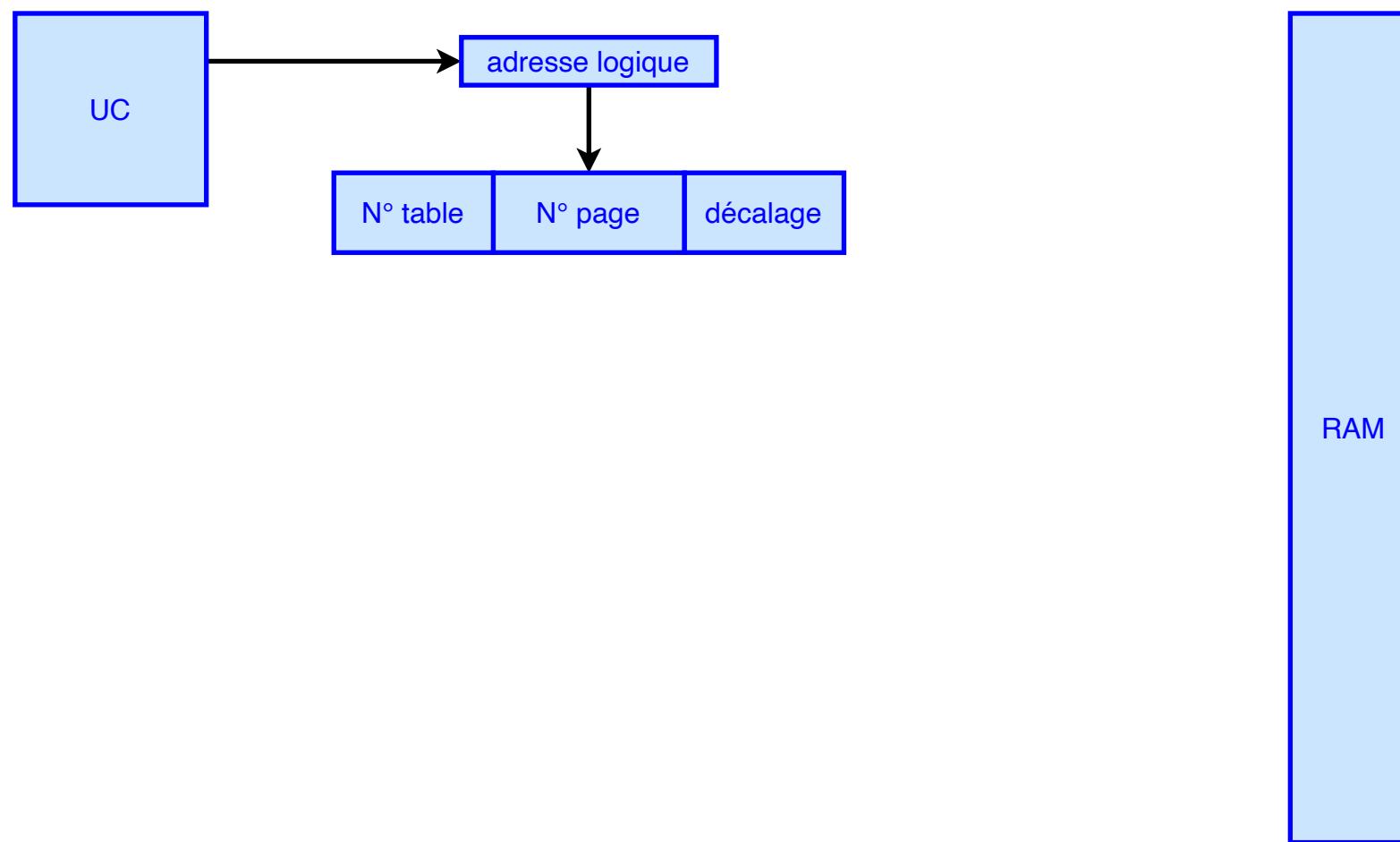
- Paginer la table des pages (et ne charger que les tables utiles)
 - Réduire la fragmentation due aux pages
 - Réduire l'espace mémoire utilisé par le système d'adressage

PAGINATION À DEUX NIVEAUX

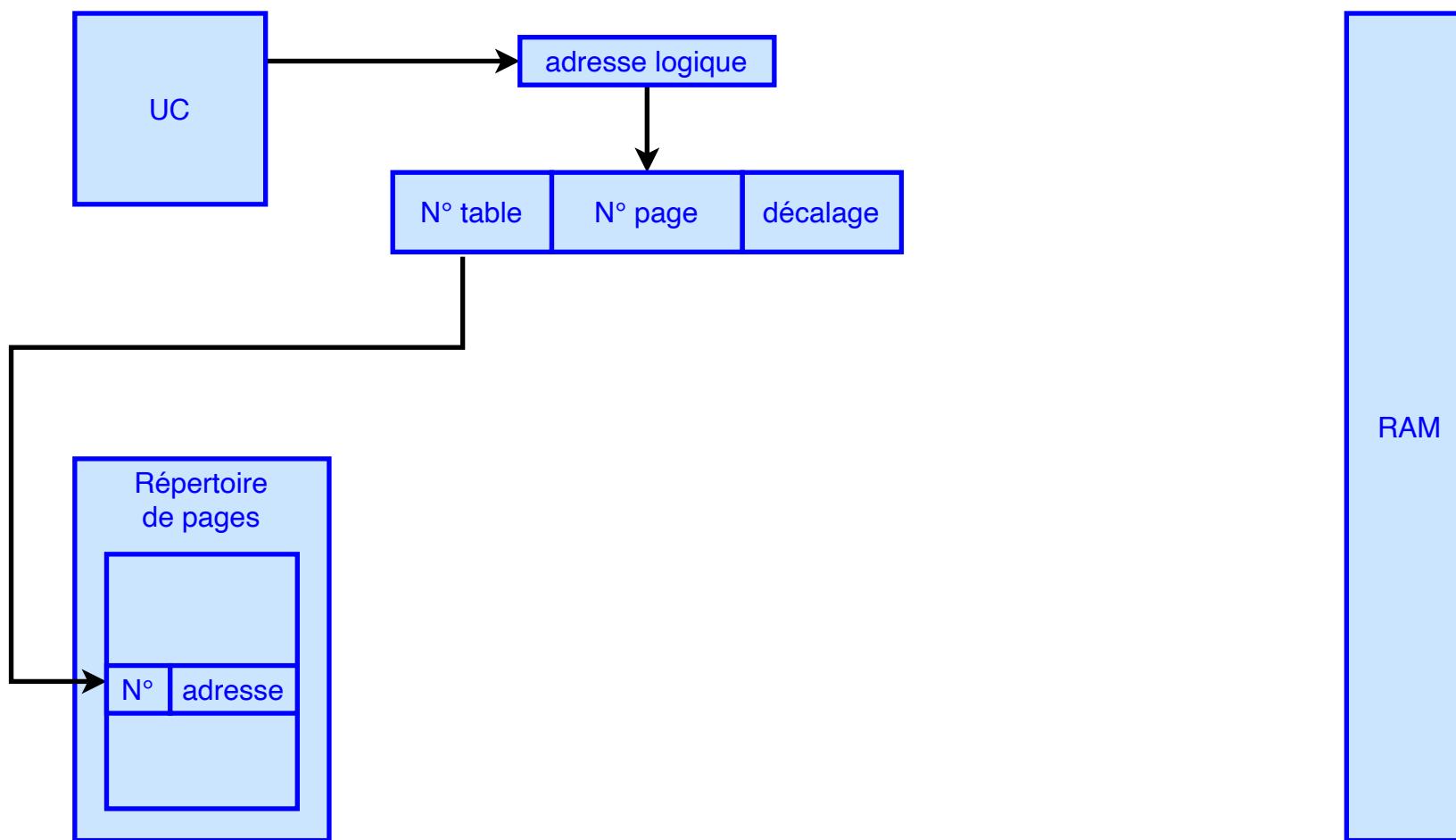
PAGINATION À DEUX NIVEAUX



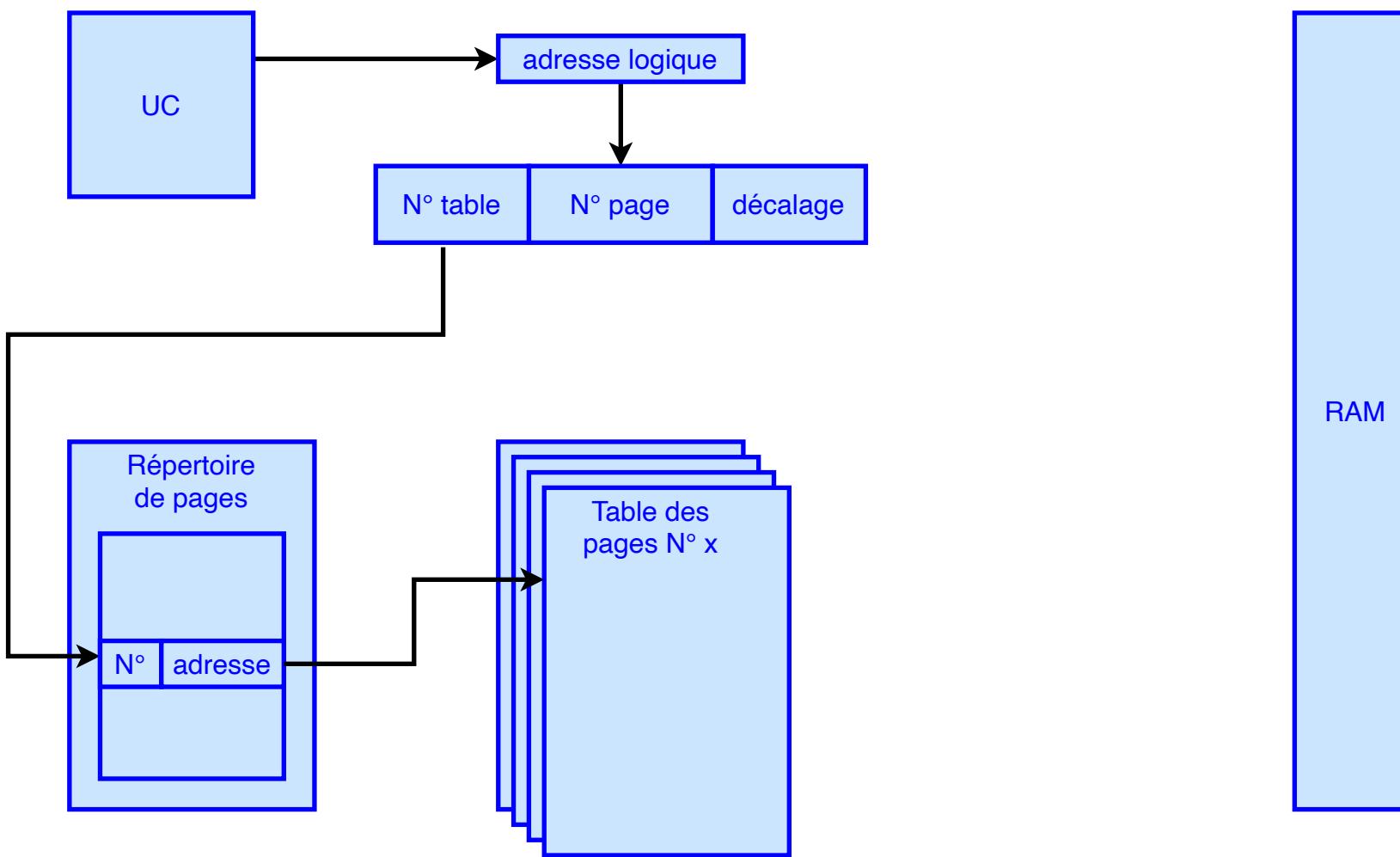
PAGINATION À DEUX NIVEAUX



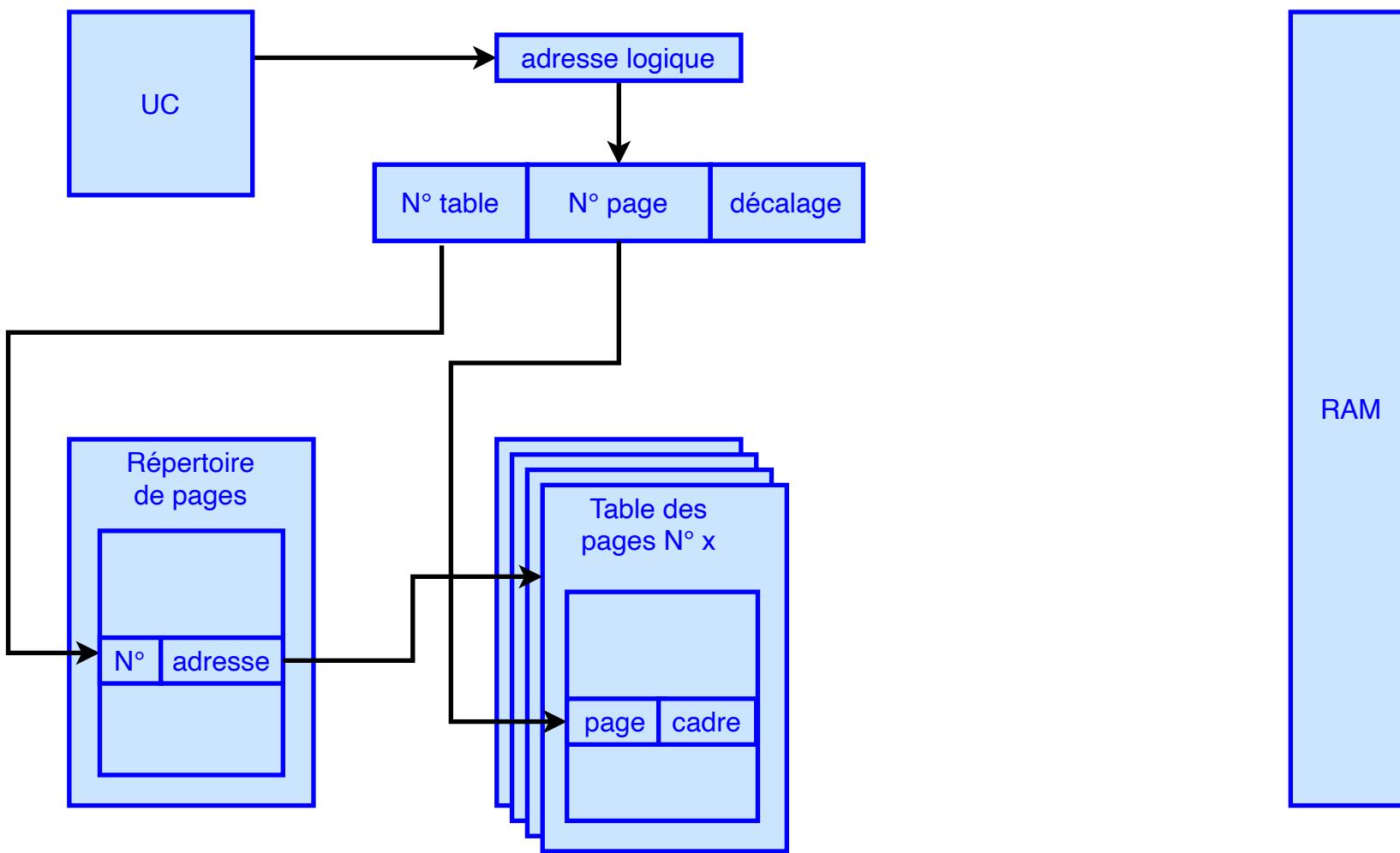
PAGINATION À DEUX NIVEAUX



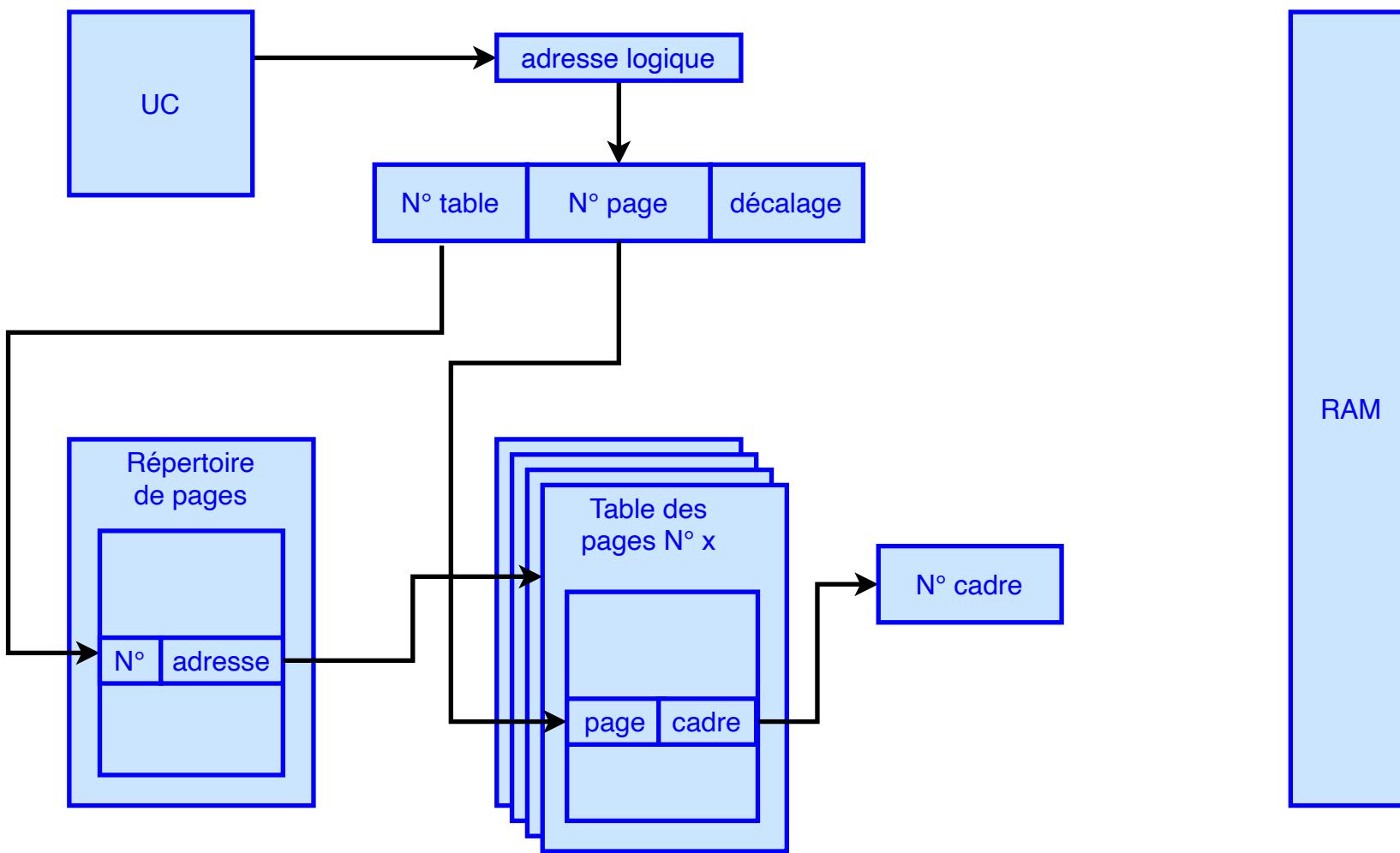
PAGINATION À DEUX NIVEAUX



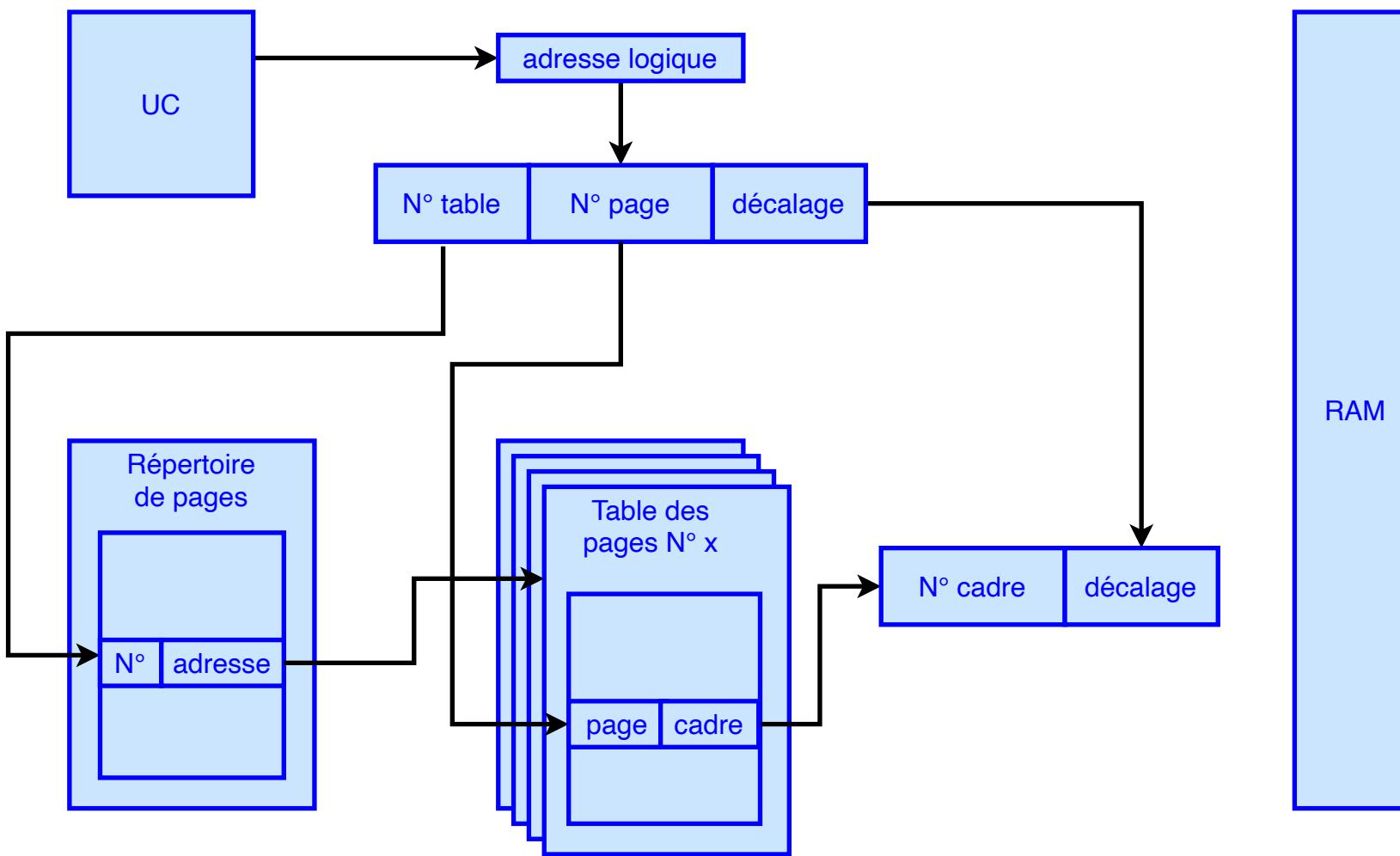
PAGINATION À DEUX NIVEAUX



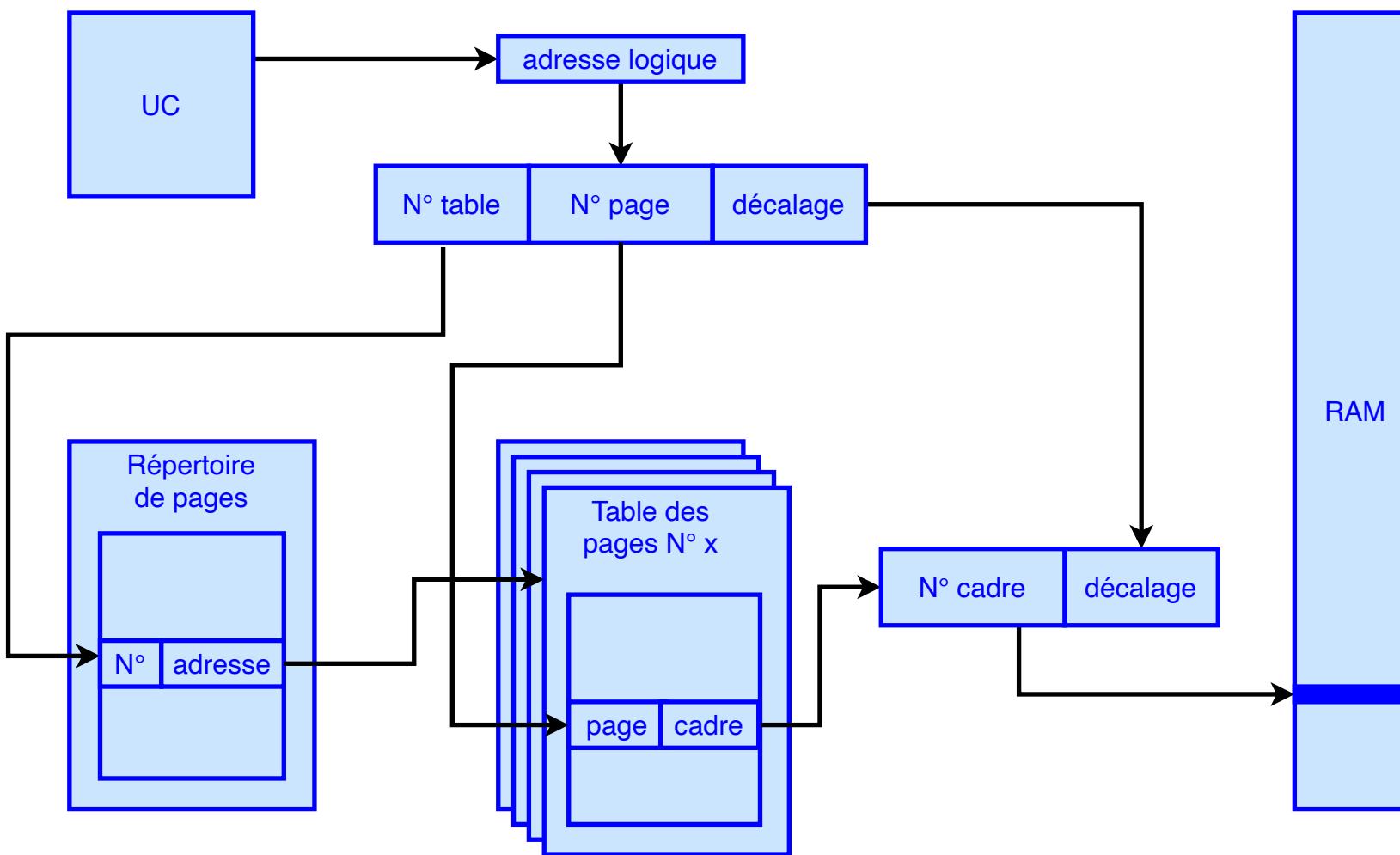
PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX



PAGINATION À DEUX NIVEAUX



ALLOCATION PAR PAGINATION

ALLOCATION PAR PAGINATION

- Avantages

ALLOCATION PAR PAGINATION

- **Avantages**
 - ✓ Pas de fragmentation

ALLOCATION PAR PAGINATION

- **Avantages**
 - ✓ Pas de fragmentation
 - ✓ Il peut y avoir plus (ou moins) de blocs que de pages

ALLOCATION PAR PAGINATION

- **Avantages**

- ✓ Pas de fragmentation
- ✓ Il peut y avoir plus (ou moins) de blocs que de pages
- ✓ Un bloc peut être partagé par plusieurs processus (code, variables)

ALLOCATION PAR PAGINATION

- **Avantages**

- ✓ Pas de fragmentation
- ✓ Il peut y avoir plus (ou moins) de blocs que de pages
- ✓ Un bloc peut être partagé par plusieurs processus (code, variables)
- ✓ Les blocs peuvent être protégés (lecture seule pour les codes)

ALLOCATION PAR PAGINATION

- **Avantages**

- ✓ Pas de fragmentation
- ✓ Il peut y avoir plus (ou moins) de blocs que de pages
- ✓ Un bloc peut être partagé par plusieurs processus (code, variables)
- ✓ Les blocs peuvent être protégés (lecture seule pour les codes)
- ✓ Possibilité d'agrandir un espace mémoire

ALLOCATION PAR PAGINATION

- **Avantages**

- ✓ Pas de fragmentation
- ✓ Il peut y avoir plus (ou moins) de blocs que de pages
- ✓ Un bloc peut être partagé par plusieurs processus (code, variables)
- ✓ Les blocs peuvent être protégés (lecture seule pour les codes)
- ✓ Possibilité d'agrandir un espace mémoire

- **Inconvénients**

- ✗ Plus coûteux

MÉMOIRE VIRTUELLE

⌚ Fonctionnalités principales d'un OS

NOMBRE ET TAILLE DES PROCESSUS

NOMBRE ET TAILLE DES PROCESSUS

Problème

- Entre 200 et 500 processus en parallèle sur un PC

NOMBRE ET TAILLE DES PROCESSUS

Problème

- Entre 200 et 500 processus en parallèle sur un PC
- **Gros processus:** Eclipse = 250 Mo, données d'un jeu \geq 1 Go

NOMBRE ET TAILLE DES PROCESSUS

Problème

- Entre 200 et 500 processus en parallèle sur un PC
- **Gros processus:** Eclipse = 250 Mo, données d'un jeu \geq 1 Go

👉 Somme des tailles des processus \geq Capacité RAM

PORTIONS DE CODE INUTILISÉES

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé
- Données du jeu → pas tout en même temps

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé
- Données du jeu → pas tout en même temps
- Données d'un tableau → pas tout en même temps

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé
- Données du jeu → pas tout en même temps
- Données d'un tableau → pas tout en même temps
- Bibliothèque → très variable!

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé
- Données du jeu → pas tout en même temps
- Données d'un tableau → pas tout en même temps
- Bibliothèque → très variable!

- **Solution**

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé
- Données du jeu → pas tout en même temps
- Données d'un tableau → pas tout en même temps
- Bibliothèque → très variable!

- **Solution**

👉 ne charger que les **pages utiles** (laisser le reste sur le disque)

PORTIONS DE CODE INUTILISÉES

- Traitement d'erreur → rarement utilisé
- Données du jeu → pas tout en même temps
- Données d'un tableau → pas tout en même temps
- Bibliothèque → très variable!

• Solution

- 👉 ne charger que les **pages utiles** (laisser le reste sur le disque)
- 👉 la **mémoire physique** sera un **cache** pour la **mémoire virtuelle**

MÉMOIRE VIRTUELLE vs PAGINATION

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient
 - un **bit de la validité** qui indique si la page est en RAM

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)
 - l'**information** pour la trouver sur le système de **stockage secondaire**

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)
 - l'**information** pour la trouver sur le système de **stockage secondaire**
- Lorsqu'un **accès à une page** non présente en RAM est fait

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)
 - l'**information** pour la trouver sur le système de **stockage secondaire**
- Lorsqu'un **accès à une page** non présente en RAM est fait
 - levée d'une **exception** CPU (**Page-Fault**), gérée par l'OS

MÉMOIRE VIRTUELLE vs PAGINATION

- Extension des mécanismes de pagination
 - la mémoire paginée peut représenter des **espaces non présents en RAM**, (en **mémoire auxiliaire**) : **mémoire virtuelle**
 - les pages sont soit en **RAM**, soit en **mémoire auxiliaire** (notion de **swap**)
- Chaque ligne de **la table des pages** contient
 - un **bit de la validité** qui indique si la page est en RAM
 - l'adresse correspondante en RAM (**numéro du bloc**)
 - l'**information** pour la trouver sur le système de **stockage secondaire**
- Lorsqu'un **accès à une page** non présente en RAM est fait
 - levée d'une **exception** CPU (**Page-Fault**), gérée par l'OS
 - mise en état bloqué du processus et chargement de la page disque en RAM

SCHÉMA DU PRINCIPE

SCHÉMA DU PRINCIPE

Mémoire virtuelle : chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

SCHÉMA DU PRINCIPE

Mémoire virtuelle : chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

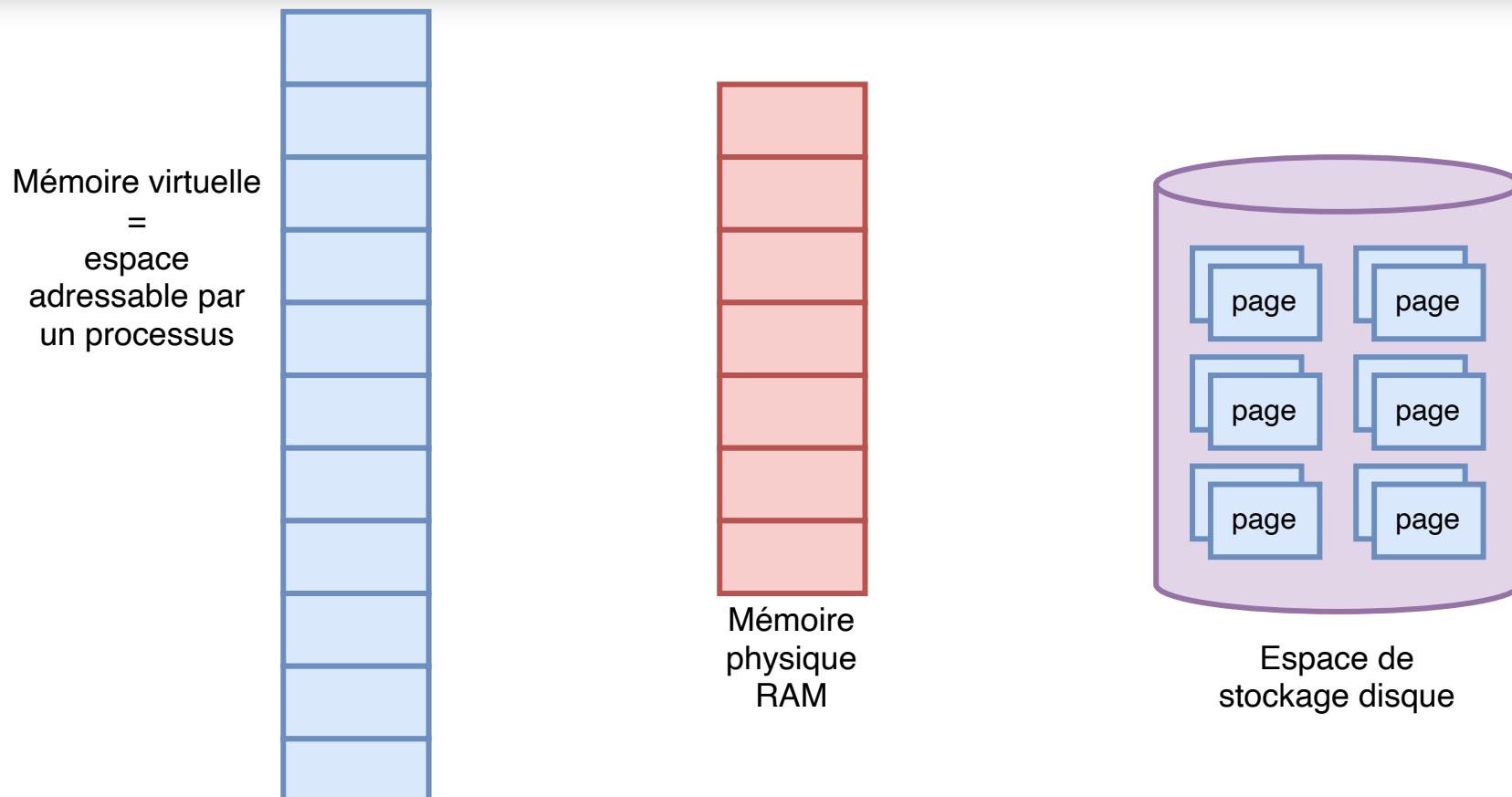


SCHÉMA DU PRINCIPE

Mémoire virtuelle : chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

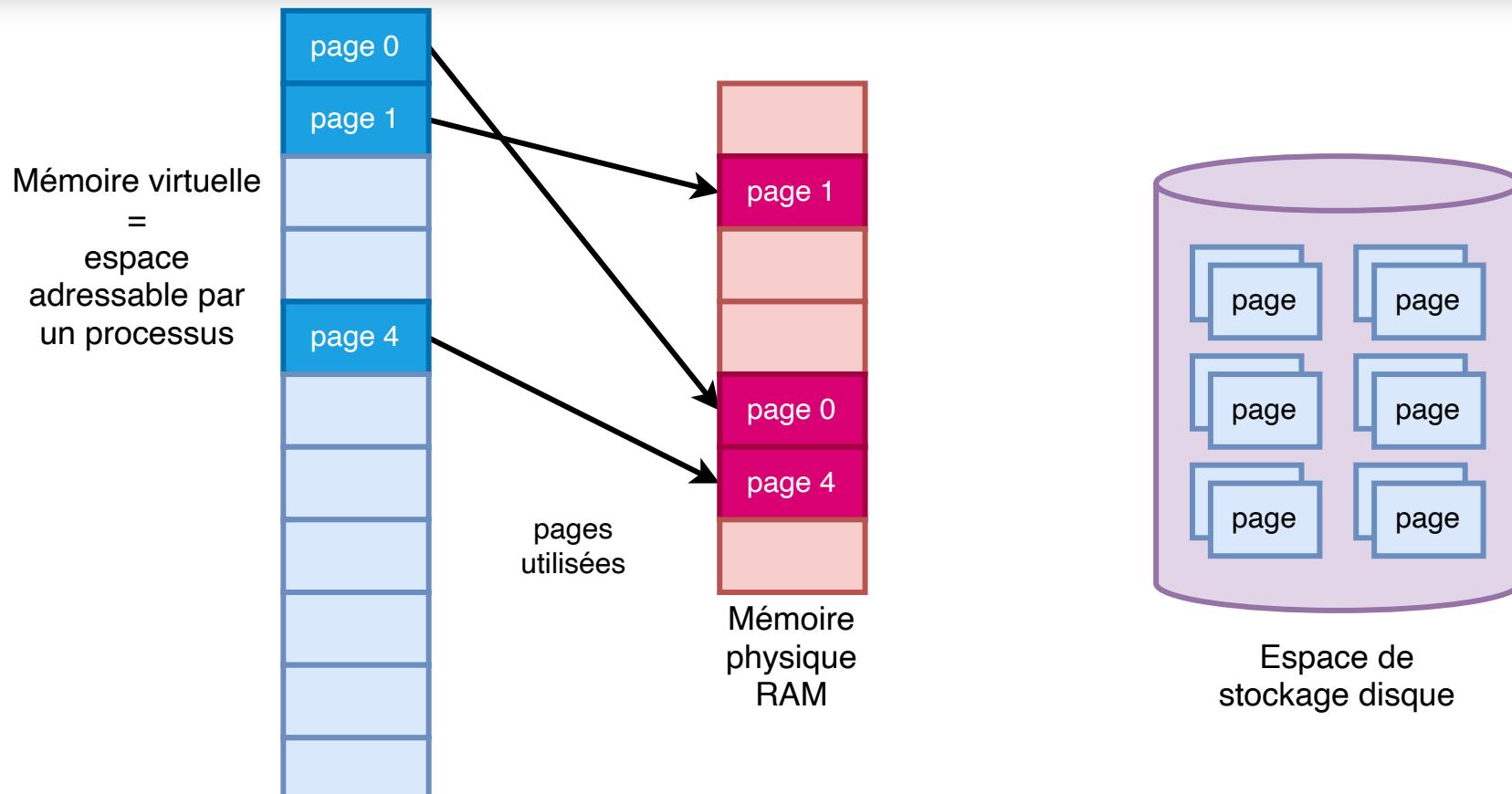
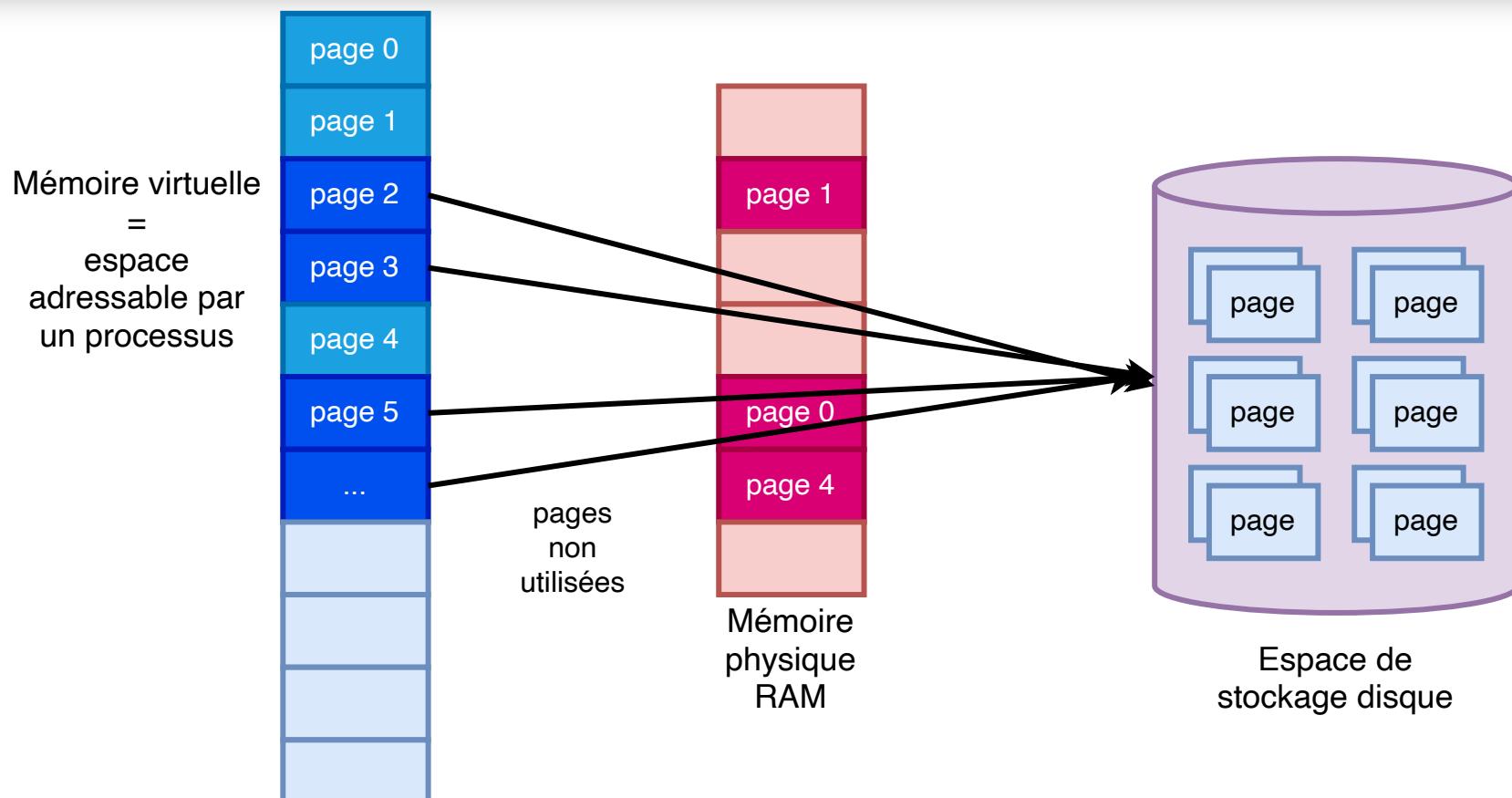


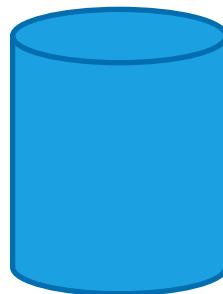
SCHÉMA DU PRINCIPE

Mémoire virtuelle : chaque processus peut **adresser** plus d'espace qu'il n'a effectivement en **mémoire physique**

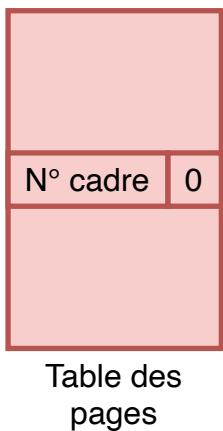
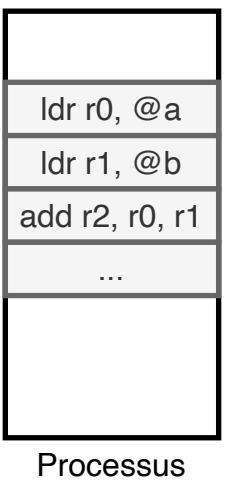


PAGINATION À LA DEMANDE

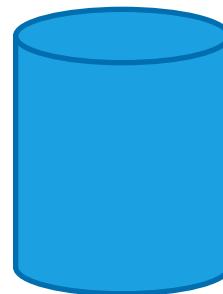
PAGINATION À LA DEMANDE



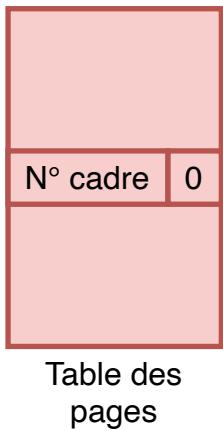
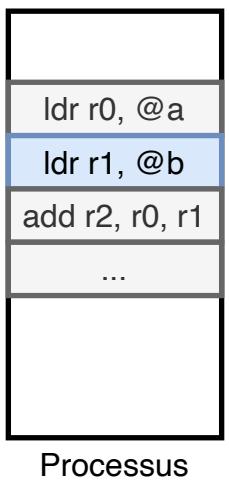
Disque



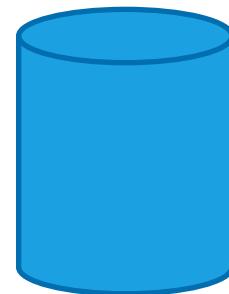
PAGINATION À LA DEMANDE



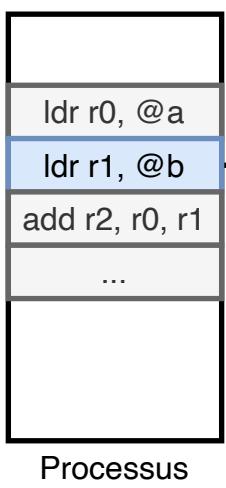
Disque



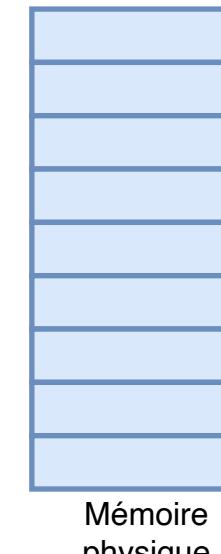
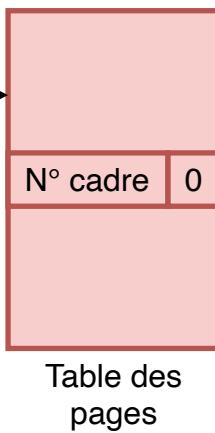
PAGINATION À LA DEMANDE



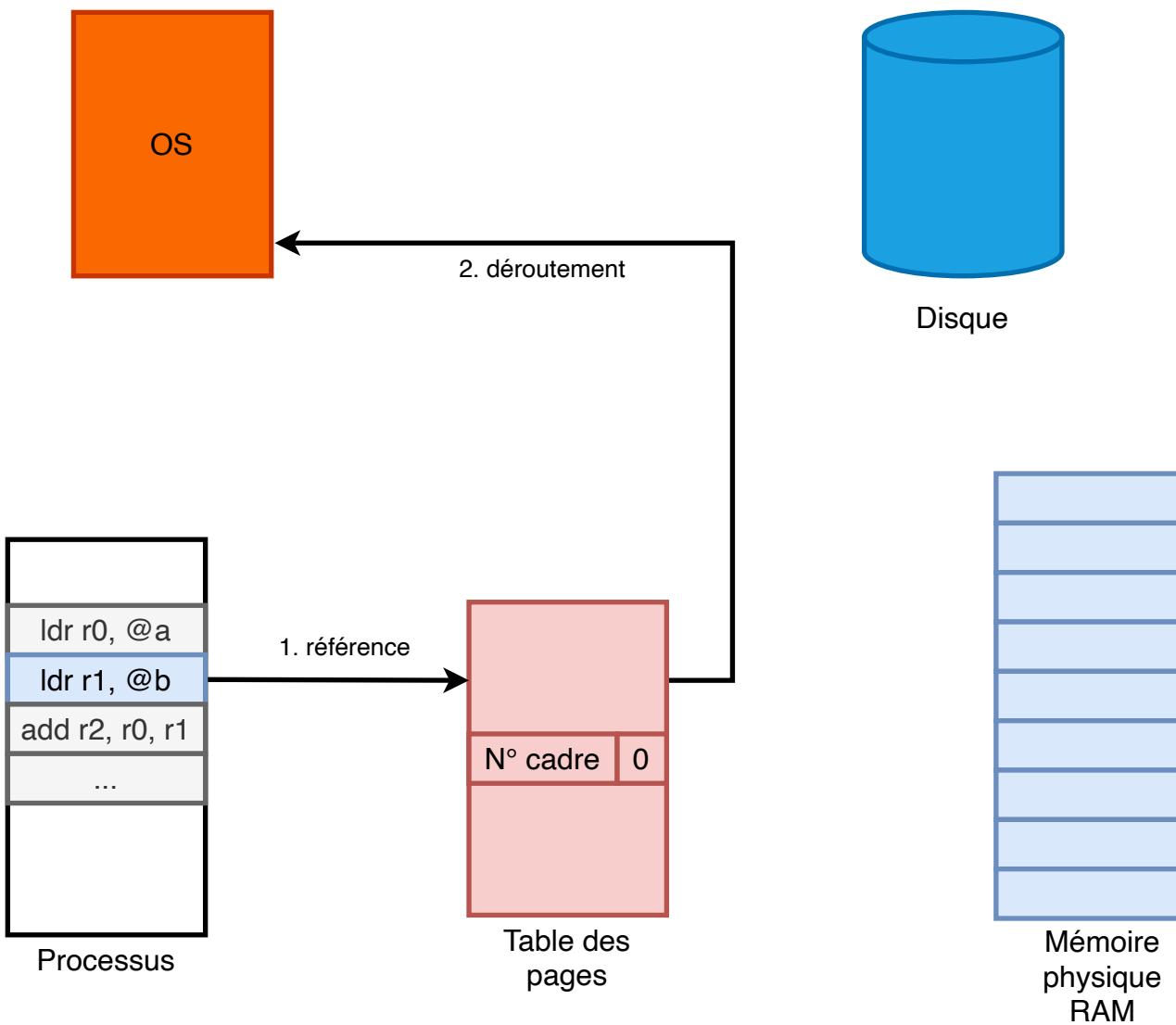
Disque



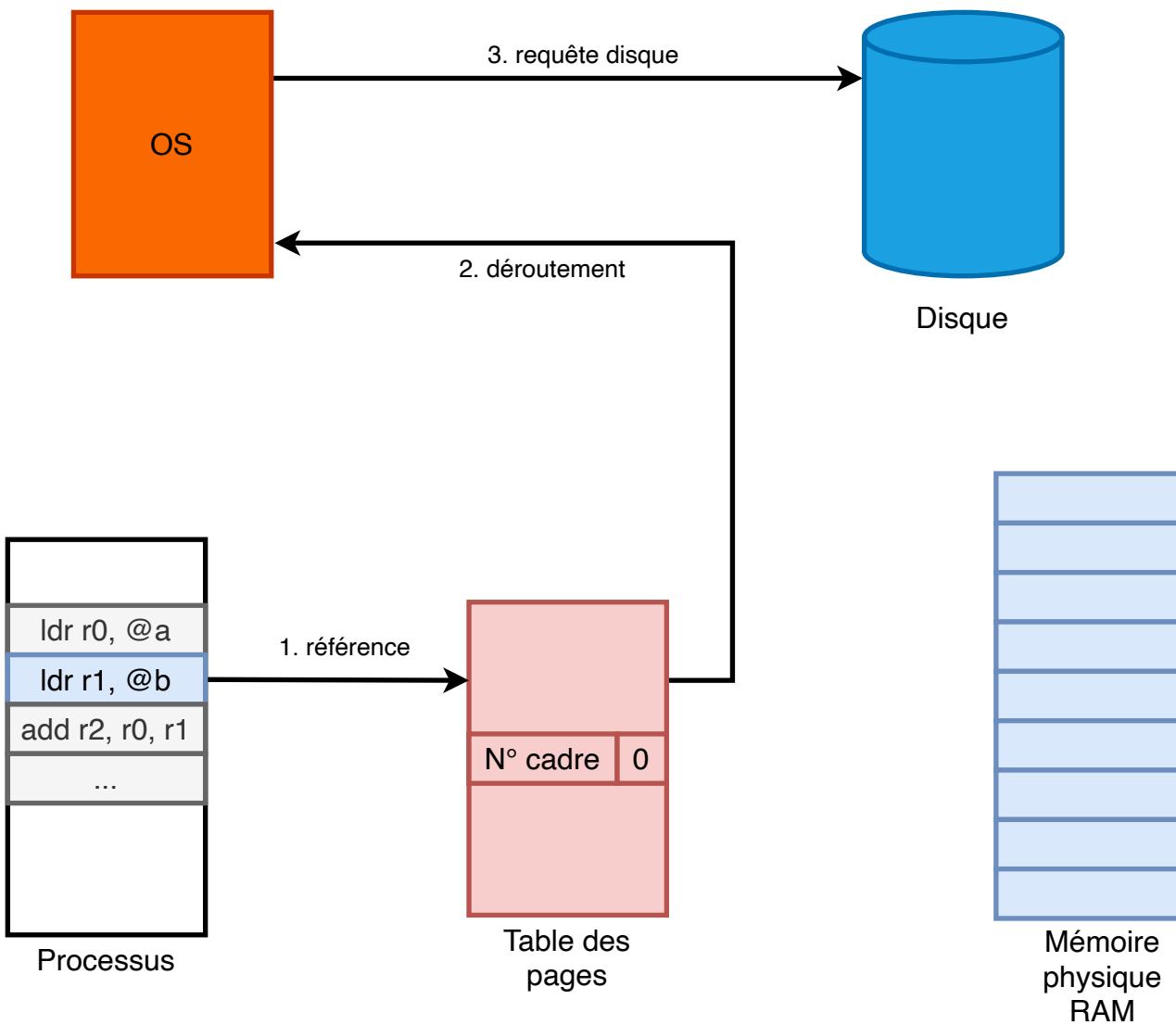
1. référence



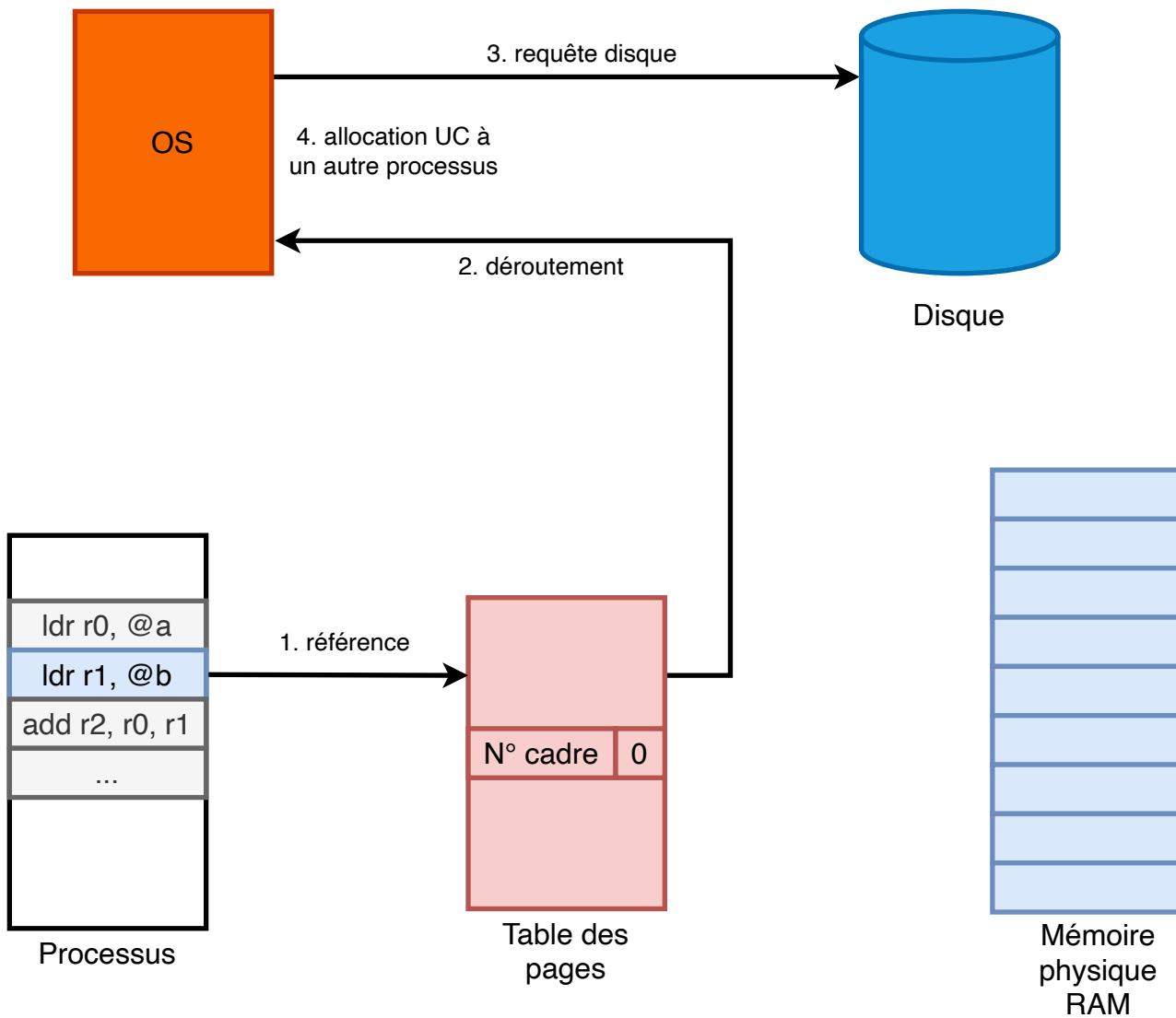
PAGINATION À LA DEMANDE



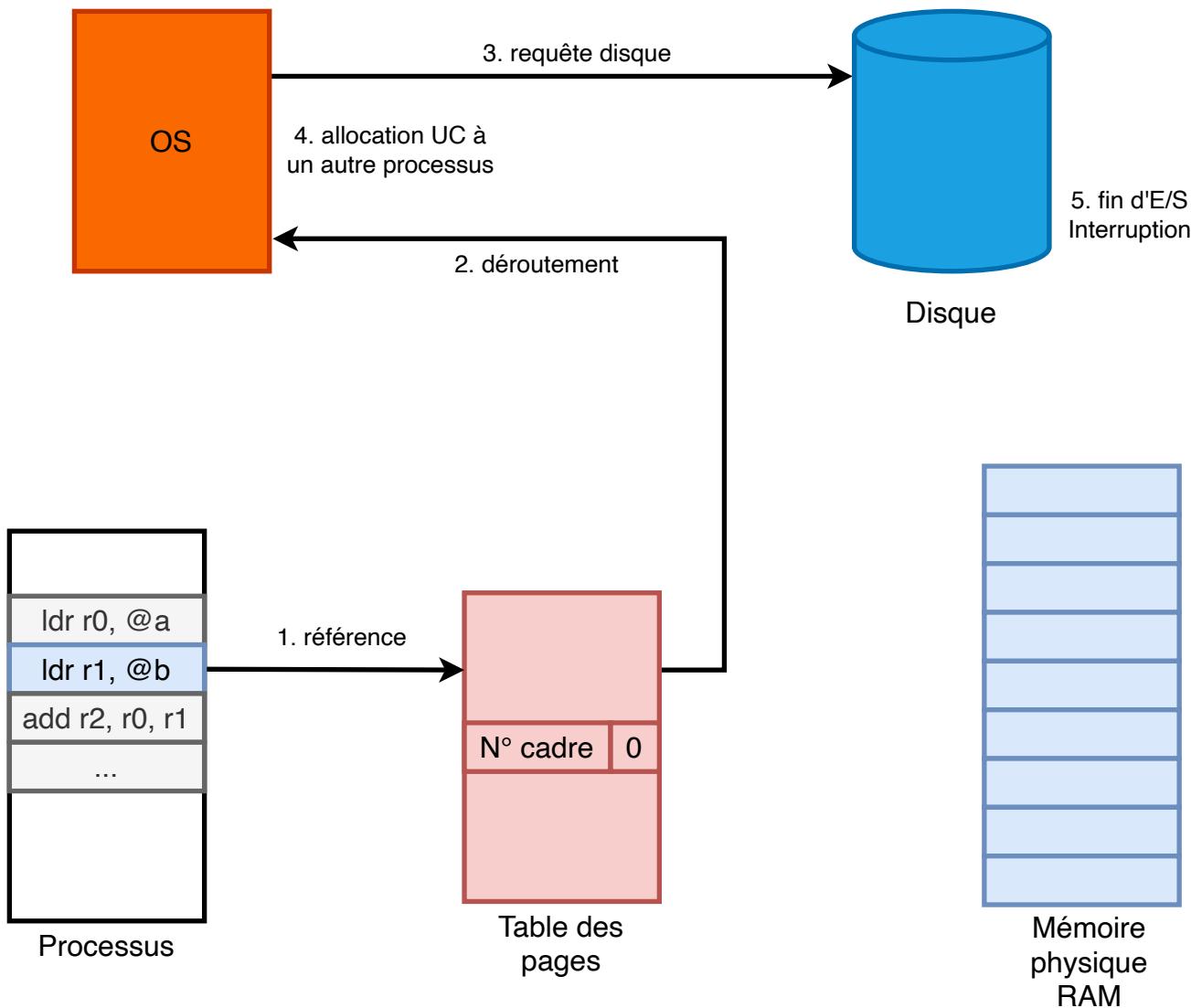
PAGINATION À LA DEMANDE



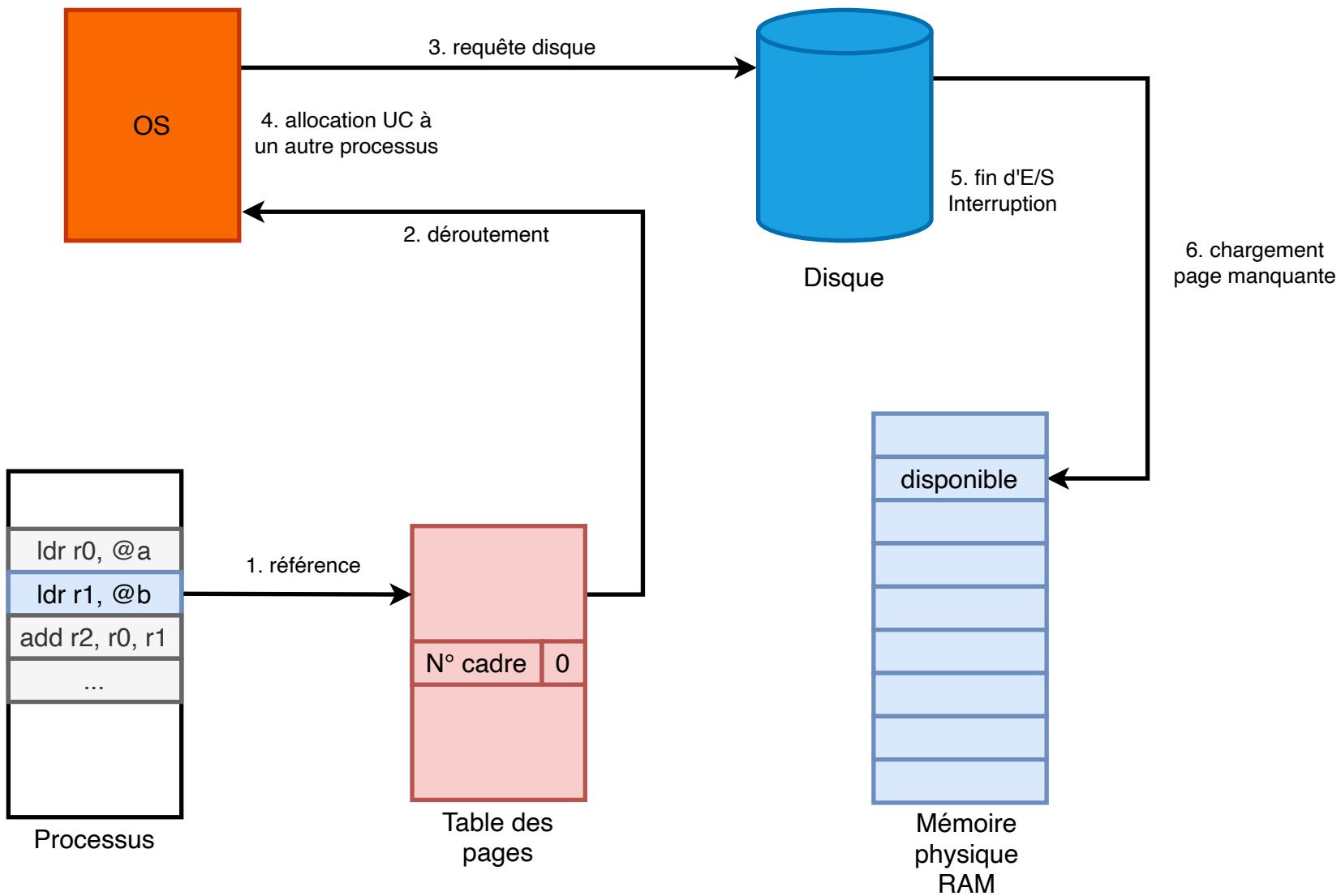
PAGINATION À LA DEMANDE



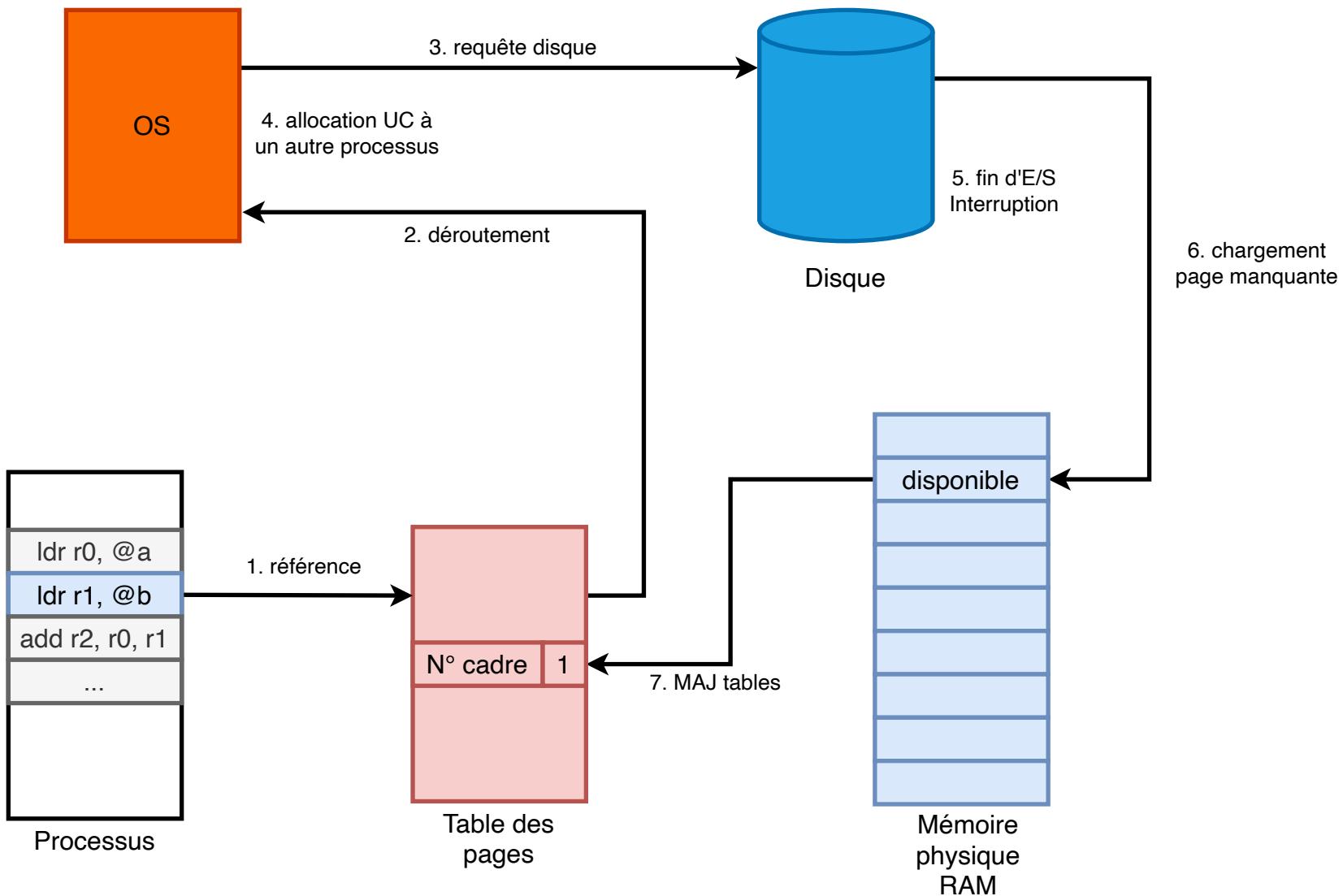
PAGINATION À LA DEMANDE



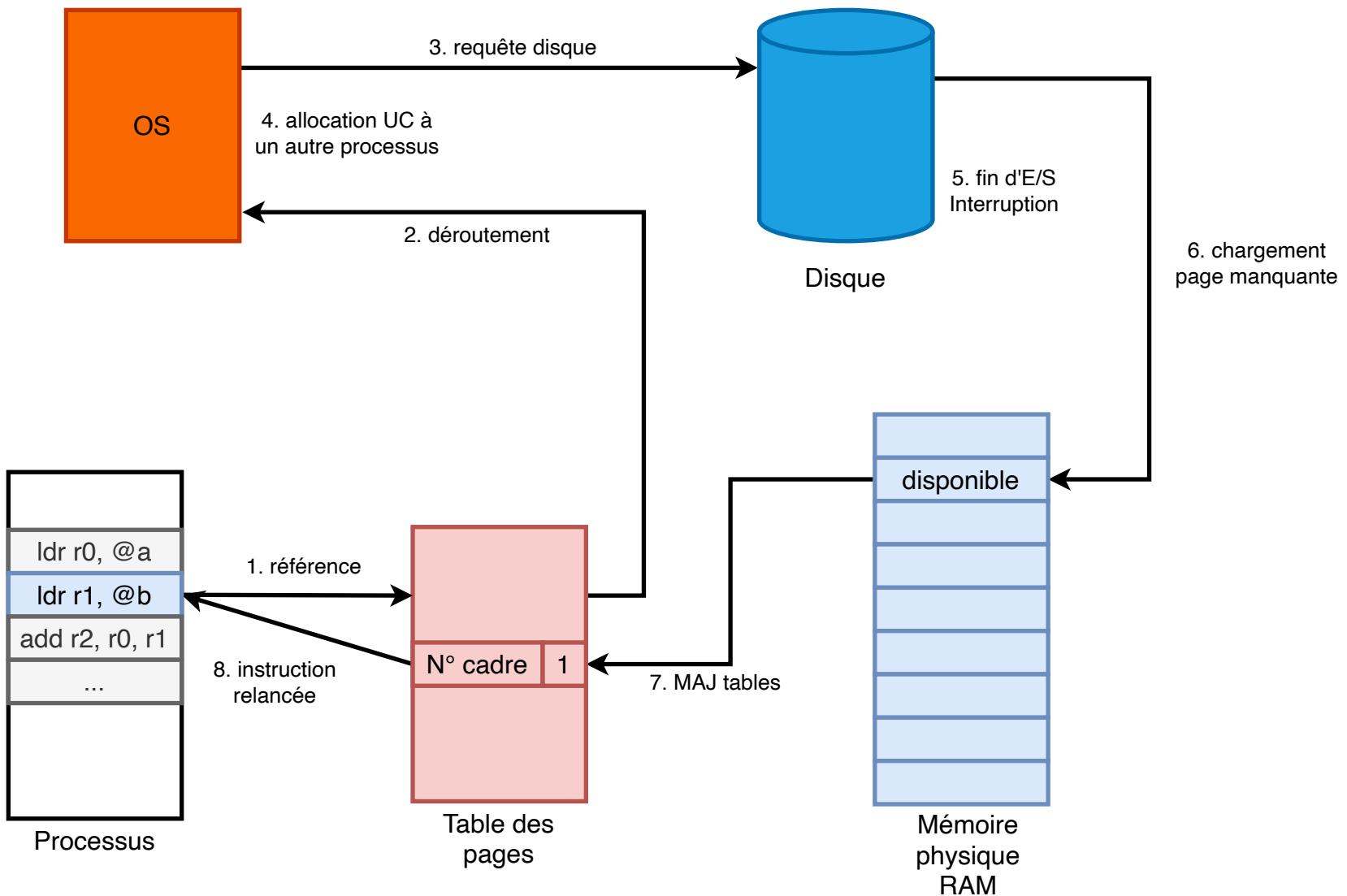
PAGINATION À LA DEMANDE



PAGINATION À LA DEMANDE



PAGINATION À LA DEMANDE



REMPLACEMENT DES PAGES

REEMPLACEMENT DES PAGES

Mémoire virtuelle

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**
👉 selon la politique d'allocation

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**
 - 👉 selon la politique d'allocation
 - 👉 libérer un cadre lorsqu'on a besoin d'une nouvelle page

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**
 - 👉 selon la politique d'allocation
 - 👉 libérer un cadre lorsqu'on a besoin d'une nouvelle page

Problème

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**
 - 👉 selon la politique d'allocation
 - 👉 libérer un cadre lorsqu'on a besoin d'une nouvelle page

Problème

- **Temps d'exécution** proportionnel au **nombre de défaut de page**

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**
 - 👉 selon la politique d'allocation
 - 👉 libérer un cadre lorsqu'on a besoin d'une nouvelle page

Problème

- **Temps d'exécution** proportionnel au **nombre de défaut de page**
 - 👉 réduire le nombre de défauts de page

REEMPLACEMENT DES PAGES

Mémoire virtuelle

- Chaque processus dispose d'un **nombre de cadres limité**
 - 👉 selon la politique d'allocation
 - 👉 libérer un cadre lorsqu'on a besoin d'une nouvelle page

Problème

- **Temps d'exécution** proportionnel au **nombre de défaut de page**
 - 👉 réduire le nombre de défauts de page
- Quelle **stratégie** pour choisir les pages à supprimer de la RAM ?

REEMPLACEMENT DES PAGES

Algorithme de remplacement

Définir une **fonction** qui, étant donné l'état actuel (occupation des cadres par des pages), décide **quel cadre doit être libéré**.

CLASSES D'ALGORITHMES

CLASSES D'ALGORITHMES

- First In, First Out (FIFO)

CLASSES D'ALGORITHMES

- **First In, First Out (FIFO)**
 - retirer les pages les plus anciennes

CLASSES D'ALGORITHMES

- **First In, First Out (FIFO)**
 - retirer les pages les plus anciennes
 - seconde chance

CLASSES D'ALGORITHMES

- **First In, First Out (FIFO)**
 - retirer les pages les plus anciennes
 - seconde chance
- **Basés sur l'utilisation des pages**

CLASSES D'ALGORITHMES

- **First In, First Out (FIFO)**
 - retirer les pages les plus anciennes
 - seconde chance
- **Basés sur l'utilisation des pages**
 - la page la moins utilisée (**Least Frequently Used**)

CLASSES D'ALGORITHMES

- **First In, First Out (FIFO)**
 - retirer les pages les plus anciennes
 - seconde chance
- **Basés sur l'utilisation des pages**
 - la page la moins utilisée (**Least Frequently Used**)
 - une page pas récemment utilisée (**Not Recently Used**)

CLASSES D'ALGORITHMES

- **First In, First Out (FIFO)**
 - retirer les pages les plus anciennes
 - seconde chance
- **Basés sur l'utilisation des pages**
 - la page la moins utilisée (**Least Frequently Used**)
 - une page pas récemment utilisée (**Not Recently Used**)
 - la page la moins récemment utilisée (**Least Recently Used**)

LIMITES DE LA MÉMOIRE PAGINÉE

LIMITES DE LA MÉMOIRE PAGINÉE

- Découpage des processus

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✗ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✗ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)
 - ✗ Fragmentation résiduelle

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✗ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)
 - ✗ Fragmentation résiduelle
- **Chargement d'une page**

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✗ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)
 - ✗ Fragmentation résiduelle
- **Chargement d'une page**
 - ✗ Plein de données inutiles

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✗ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)
 - ✗ Fragmentation résiduelle
- **Chargement d'une page**
 - ✗ Plein de données inutiles
 - ✗ Pas forcément tout ce dont on a besoin

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✖ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)
 - ✖ Fragmentation résiduelle
- **Chargement d'une page**
 - ✖ Plein de données inutiles
 - ✖ Pas forcément tout ce dont on a besoin
- **Idée : mémoire segmentée** (non traitée dans ce cours)

LIMITES DE LA MÉMOIRE PAGINÉE

- **Découpage des processus**
 - ✖ Taille arbitraire ($2^{nb_bits_cadre}$)
(peut couper une portion de code, un bloc de données ...)
 - ✖ Fragmentation résiduelle
- **Chargement d'une page**
 - ✖ Plein de données inutiles
 - ✖ Pas forcément tout ce dont on a besoin
- **Idée : mémoire segmentée** (non traitée dans ce cours)
 - Découper en tenant compte de la structure du processus
(code + données)

SYSTÈME DE FICHIERS

⌚ Fonctionnalités principales d'un OS

LE STOCKAGE SECONDAIRE

LE STOCKAGE SECONDAIRE

- **Le stockage secondaire** conserve programmes et données

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:
 - une vue logique des données sous la forme de :

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:
 - une vue logique des données sous la forme de :
 - 👉 **fichiers** (files) : unité de stockage logique
 - 👉 **répertoires** : classement arborescent
 - 👉 **volumes montés** : vue globale des systèmes de fichiers

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:
 - une vue logique des données sous la forme de :
 - 👉 **fichiers** (files) : unité de stockage logique
 - 👉 **répertoires** : classement arborescent
 - 👉 **volumes montés** : vue globale des systèmes de fichiers
 - une organisation physique des espaces de stockage

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:
 - une vue logique des données sous la forme de :
 - 👉 **fichiers** (files) : unité de stockage logique
 - 👉 **répertoires** : classement arborescent
 - 👉 **volumes montés** : vue globale des systèmes de fichiers
 - une organisation physique des espaces de stockage
 - 👉 découpage en blocs
 - 👉 affectation et libération de blocs

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:
 - une vue logique des données sous la forme de :
 - 👉 **fichiers** (files) : unité de stockage logique
 - 👉 **répertoires** : classement arborescent
 - 👉 **volumes montés** : vue globale des systèmes de fichiers
 - une organisation physique des espaces de stockage
 - 👉 découpage en blocs
 - 👉 affectation et libération de blocs
 - un **système d'entrées/sorties** (non traité dans ce cours)

LE STOCKAGE SECONDAIRE

- Le stockage secondaire conserve programmes et données
- L'OS a pour objectif de masquer la complexité et la diversité des unités de stockage (matériel, système de fichiers) au travers:
 - une vue logique des données sous la forme de :
 - 👉 **fichiers** (files) : unité de stockage logique
 - 👉 **répertoires** : classement arborescent
 - 👉 **volumes montés** : vue globale des systèmes de fichiers
 - une organisation physique des espaces de stockage
 - 👉 découpage en blocs
 - 👉 affectation et libération de blocs
 - un système d'entrées/sorties (non traité dans ce cours)
 - 👉 gestion de caches, algorithmes d'optimisation des accès disques,
 - 👉 pilotes gérant les communications avec les périphériques

LA VUE LOGIQUE

Fichier

- une **collection nommée** d'information accessibles via un périphérique.

LA VUE LOGIQUE

Fichier

- une **collection nommée** d'information accessibles via un périphérique.

Système de fichiers (**File System - FS**)

- un ensemble de **structures de données** et de **fonctions** permettant à un OS de manipuler des fichiers.

FILE SYSTEM - FS

FILE SYSTEM - FS

- Le **FS** (point de vue de l'OS) rend des services qui **ne dépendent pas de son implémentation** (*indépendant du support physique*).

FILE SYSTEM - FS

- Le **FS** (point de vue de l'OS) rend des services qui **ne dépendent pas de son implémentation** (*indépendant du support physique*).
- Le **FS** répond à des **problèmes** comme :

FILE SYSTEM - FS

- Le **FS** (point de vue de l'OS) rend des services qui **ne dépendent pas de son implémentation** (*indépendant du support physique*).
- Le **FS** répond à des **problèmes** comme :
 - la **diversité** des supports de stockage ;

FILE SYSTEM - FS

- Le **FS** (point de vue de l'OS) rend des services qui **ne dépendent pas de son implémentation** (*indépendant du support physique*).
- Le **FS** répond à des **problèmes** comme :
 - la **diversité** des supports de stockage ;
 - la **sécurisation** des données.

FILE CONTROL BLOCK - FCB

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom:** indépendant de l'OS, lisible

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom:** indépendant de l'OS, lisible
 - **Identifiant:** numérique, unique, pour l'OS

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom:** indépendant de l'OS, lisible
 - **Identifiant:** numérique, unique, pour l'OS
 - **Type:** extension

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom:** indépendant de l'OS, lisible
 - **Identifiant:** numérique, unique, pour l'OS
 - **Type:** extension
 - **Emplacement:** pointeur sur un périphérique

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom:** indépendant de l'OS, lisible
 - **Identifiant:** numérique, unique, pour l'OS
 - **Type:** extension
 - **Emplacement:** pointeur sur un périphérique
 - **Taille:** en octets ou en blocs

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom:** indépendant de l'OS, lisible
 - **Identifiant:** numérique, unique, pour l'OS
 - **Type:** extension
 - **Emplacement:** pointeur sur un périphérique
 - **Taille:** en octets ou en blocs
 - **Protection:** lecture, écriture, exécution...

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom**: indépendant de l'OS, lisible
 - **Identifiant**: numérique, unique, pour l'OS
 - **Type**: extension
 - **Emplacement**: pointeur sur un périphérique
 - **Taille**: en octets ou en blocs
 - **Protection**: lecture, écriture, exécution...
 - **Date(s)**: création, modification, accès...

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom**: indépendant de l'OS, lisible
 - **Identifiant**: numérique, unique, pour l'OS
 - **Type**: extension
 - **Emplacement**: pointeur sur un périphérique
 - **Taille**: en octets ou en blocs
 - **Protection**: lecture, écriture, exécution...
 - **Date(s)**: création, modification, accès...
 - **Utilisateur**: propriétaire du fichier

FILE CONTROL BLOCK - FCB

- Le **FCB** : une **structure de données** de l'OS utilisée pour stocker les *informations nécessaires à la gestion des fichiers*.
 - **Nom**: indépendant de l'OS, lisible
 - **Identifiant**: numérique, unique, pour l'OS
 - **Type**: extension
 - **Emplacement**: pointeur sur un périphérique
 - **Taille**: en octets ou en blocs
 - **Protection**: lecture, écriture, exécution...
 - **Date(s)**: création, modification, accès...
 - **Utilisateur**: propriétaire du fichier
 - ...

NOTION DE RÉPERTOIRE

NOTION DE RÉPERTOIRE

- Le **répertoire** représente la structure de **stockage des informations** des fichiers (les **FCB**) dans les **supports de stockage**.

NOTION DE RÉPERTOIRE

- Le **répertoire** représente la structure de **stockage des informations** des fichiers (les **FCB**) dans les **supports de stockage**.
👉 entrée du répertoire = identifiant du fichier et/ou nom du fichier

NOTION DE RÉPERTOIRE

- Le **répertoire** représente la structure de **stockage des informations** des fichiers (les **FCB**) dans les **supports de stockage**.
 - 👉 entrée du répertoire = identifiant du fichier et/ou nom du fichier
 - 👉 contenu du répertoire = FCB des fichiers

NOTION DE RÉPERTOIRE

- Le **répertoire** représente la structure de **stockage des informations** des fichiers (les **FCB**) dans les **supports de stockage**.
 - 👉 entrée du répertoire = identifiant du fichier et/ou nom du fichier
 - 👉 contenu du répertoire = FCB des fichiers
- L'**OS** récupère **les informations** sur les fichiers dans le répertoire

RÉPERTOIRE / DISQUES

RÉPERTOIRE / DISQUES

- **Disque** : structure physique

RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**

RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**
 - Base : 1 disque = 1 partition

RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**
 - Base : 1 disque = 1 partition
 - 1 disque = N partitions

RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**
 - Base : 1 disque = 1 partition
 - 1 disque = N partitions
 - 1 partition = 1 ou N disques (selon les OS)

RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**
 - Base : 1 disque = 1 partition
 - 1 disque = N partitions
 - 1 partition = 1 ou N disques (selon les OS)
- **Répertoire** (cas de structure à un niveau)

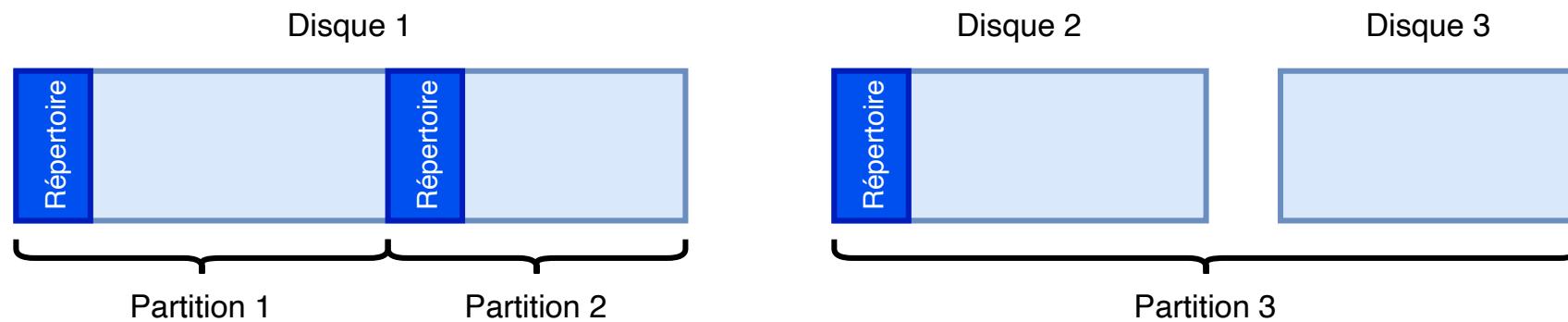
RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**
 - Base : 1 disque = 1 partition
 - 1 disque = N partitions
 - 1 partition = 1 ou N disques (selon les OS)
- **Répertoire** (cas de structure à un niveau)
 - un répertoire par partition → l'ensemble des FCB

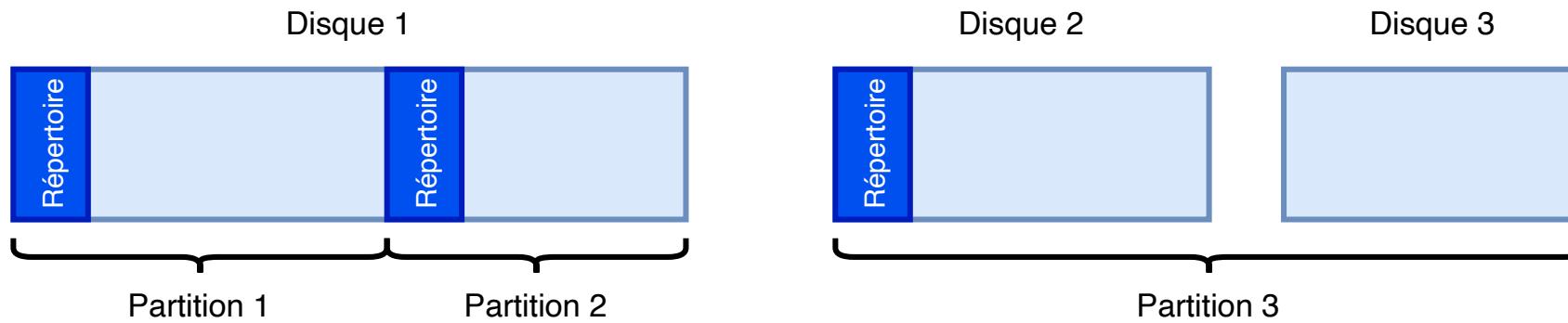
RÉPERTOIRE / DISQUES

- **Disque** : structure physique
- **Structure logique (disque "virtuel")**
 - Base : 1 disque = 1 partition
 - 1 disque = N partitions
 - 1 partition = 1 ou N disques (selon les OS)
- **Répertoire** (cas de structure à un niveau)
 - un répertoire par partition → l'ensemble des FCB
 - entrée : nom/identifiant → FCB

STRUCTURE À UN NIVEAU

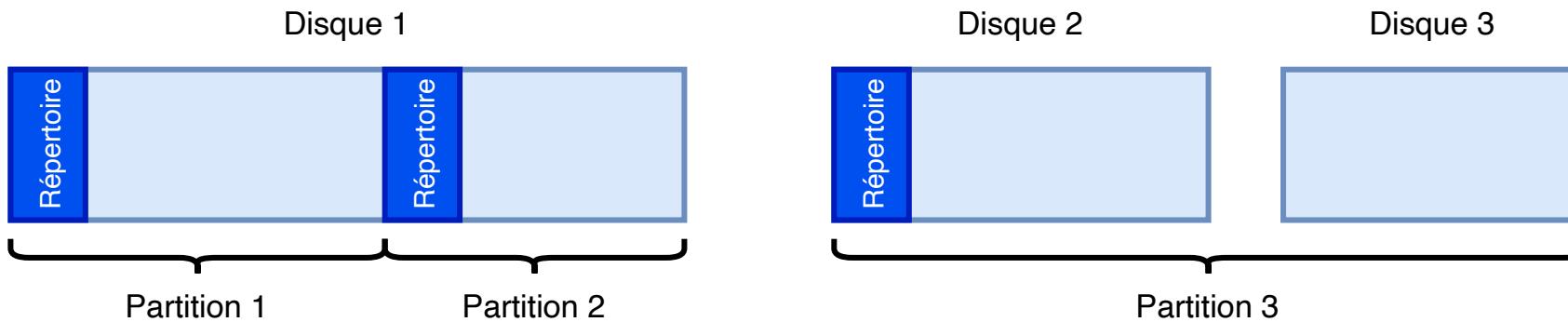


STRUCTURE À UN NIVEAU



Exemples

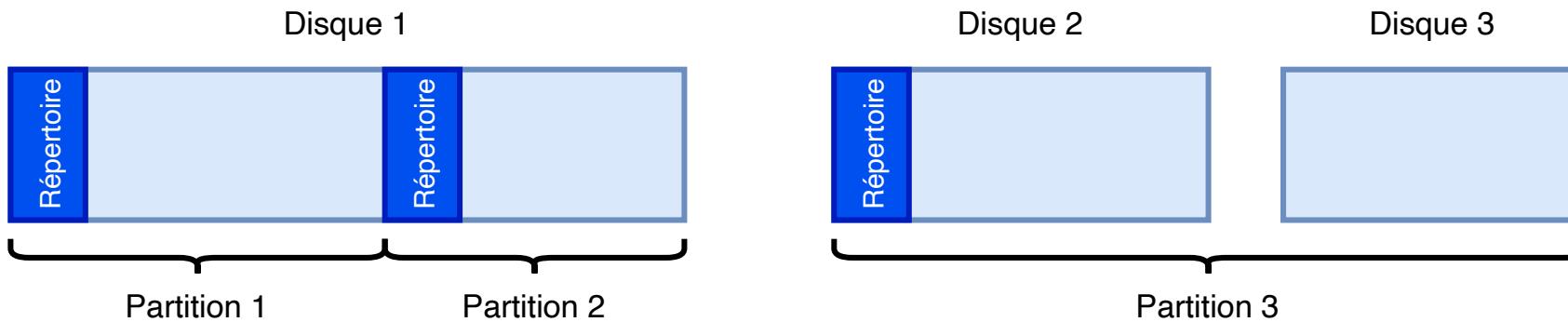
STRUCTURE À UN NIVEAU



Exemples

- **MSDOS et Windows** : Taille limitée à 11 caractères
 - 👉 8 noms + 3 extensions
 - 👉 50^{11} fichiers max par répertoire ($\approx 5.10^{18}$)

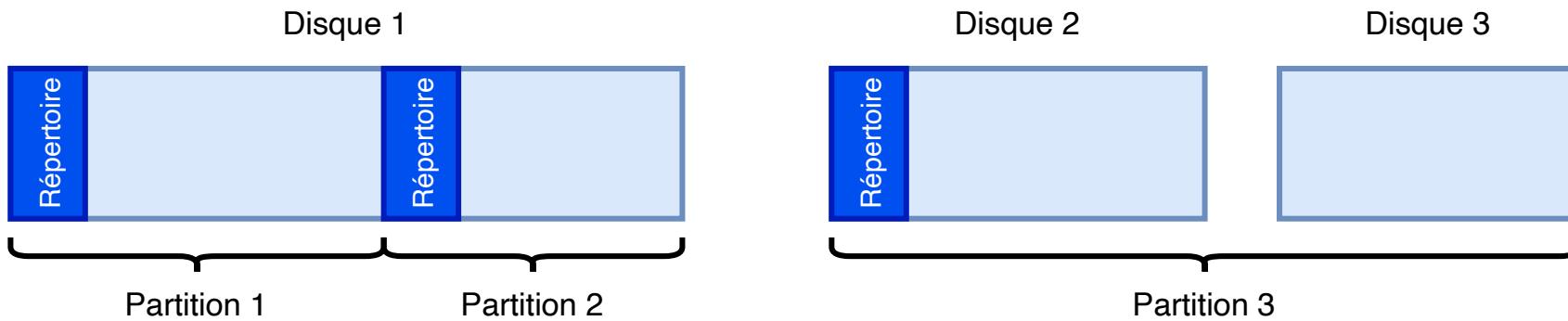
STRUCTURE À UN NIVEAU



Exemples

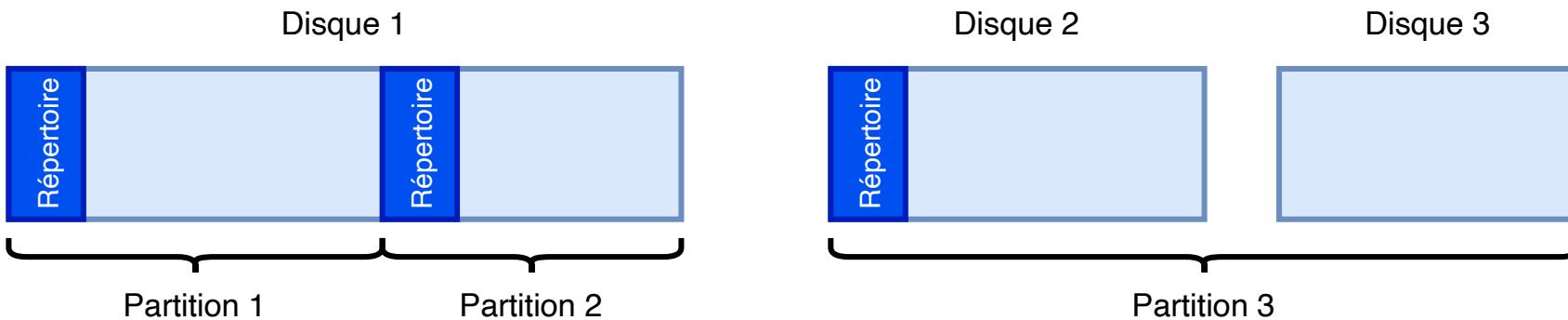
- **MSDOS et Windows** : Taille limitée à 11 caractères
 - 👉 8 noms + 3 extensions
 - 👉 50^{11} fichiers max par répertoire ($\approx 5.10^{18}$)
- **Unix et Mac** : Taille limitée à 255 octets

STRUCTURE À UN NIVEAU



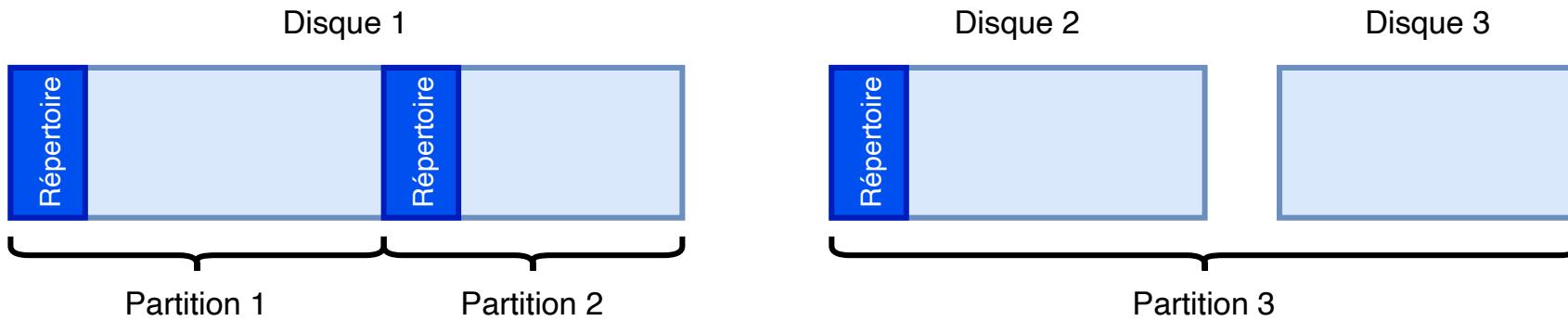
✓ Nom → FCB

STRUCTURE À UN NIVEAU



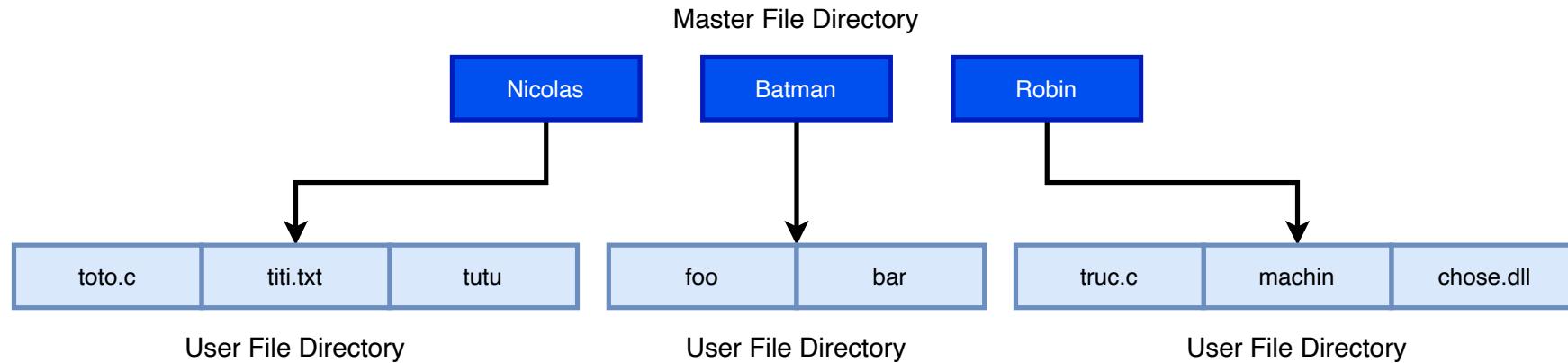
- ✗ Taille du répertoire proportionnelle au nombre de fichiers
- ✓ Nom → FCB

STRUCTURE À UN NIVEAU

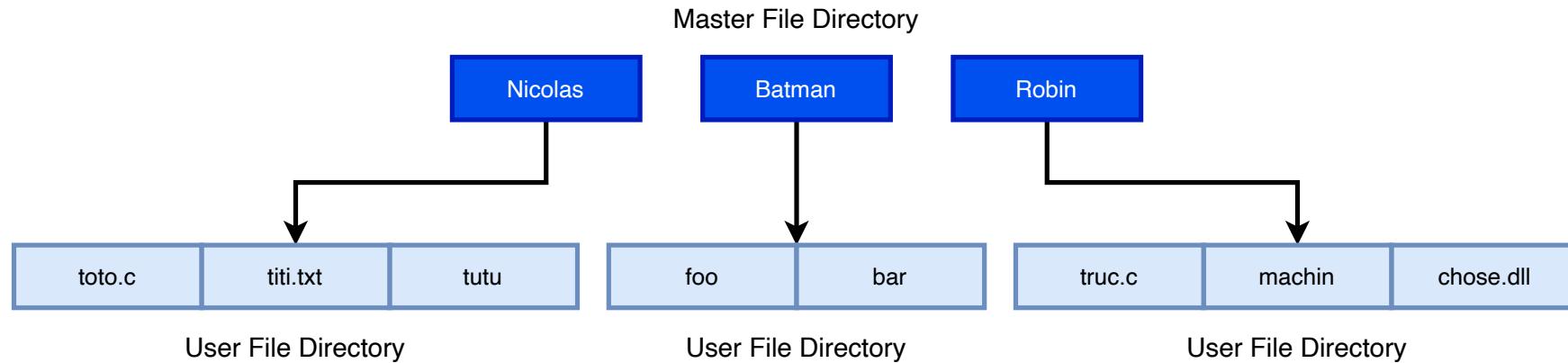


- ✗ Taille du répertoire proportionnelle au nombre de fichiers
- ✓ Nom → FCB ✗ Utilisateur : organiser les fichiers, unicité de nom, ...

STRUCTURE À DEUX NIVEAUX

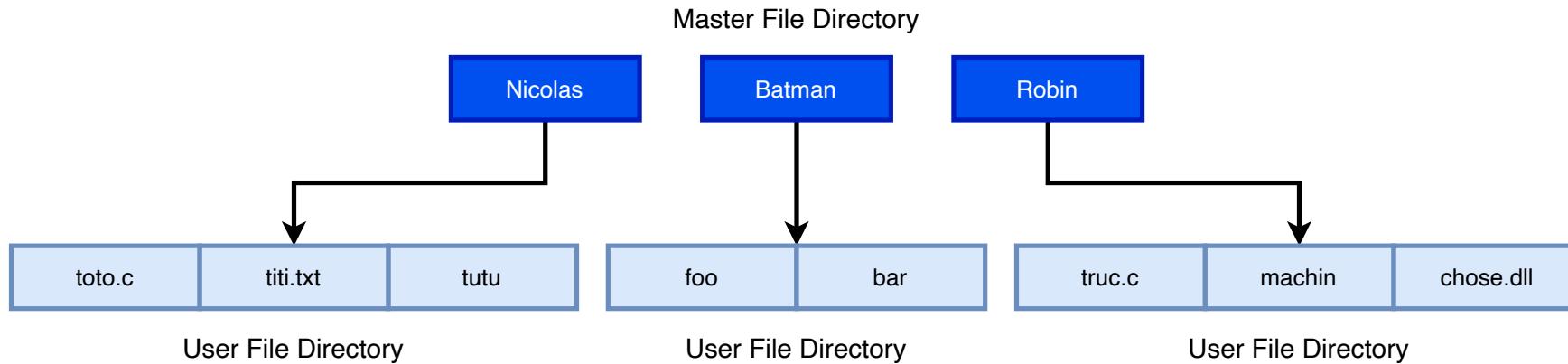


STRUCTURE À DEUX NIVEAUX



Principe

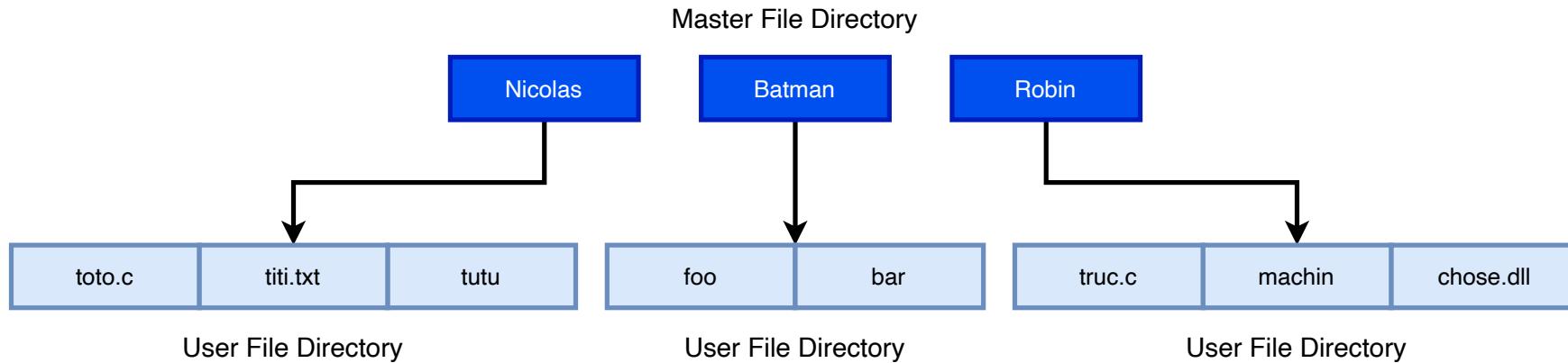
STRUCTURE À DEUX NIVEAUX



Principe

- Répertoire des utilisateurs: **Master File Directory (MFD)**
👉 Identifiant → User File Directory (UFD)

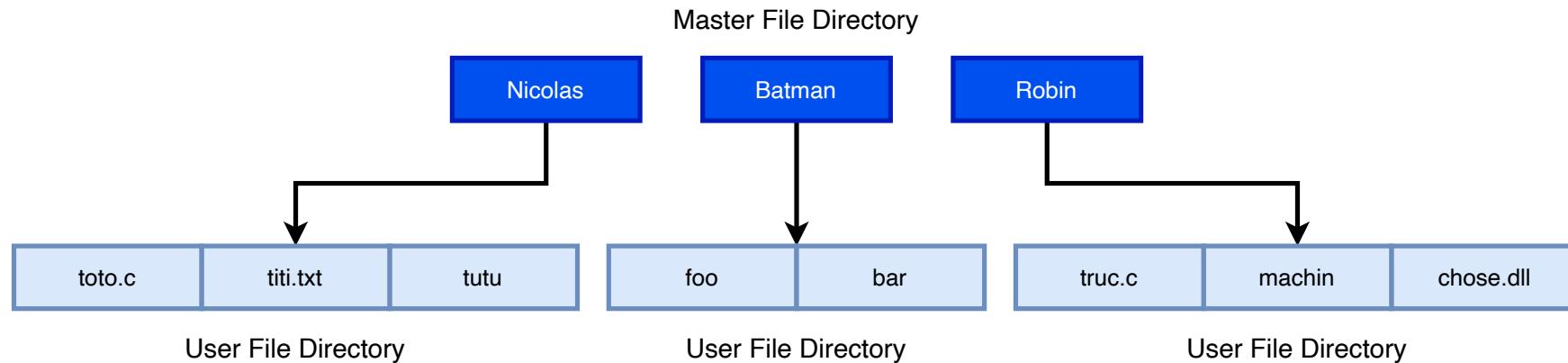
STRUCTURE À DEUX NIVEAUX



Principe

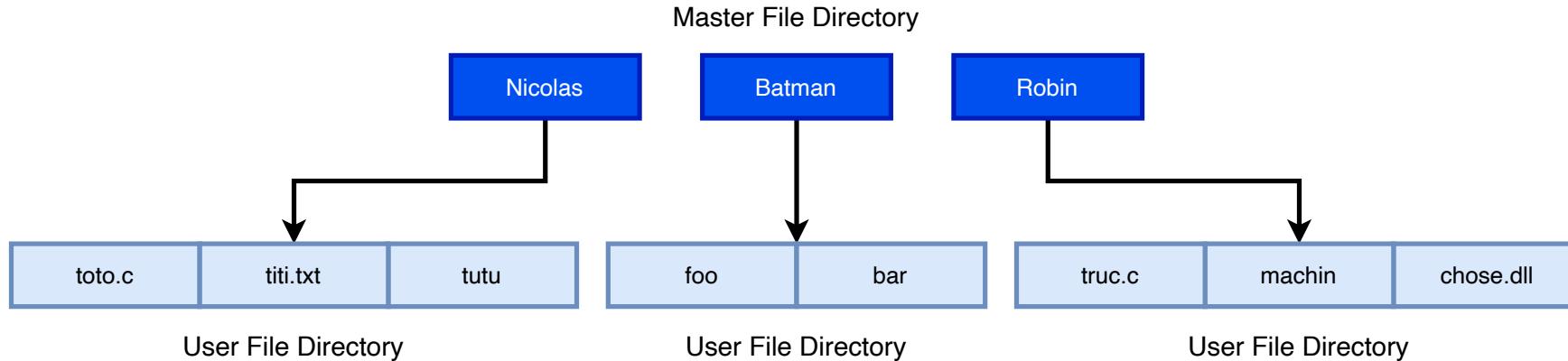
- Répertoire des utilisateurs: **Master File Directory (MFD)**
 - 👉 Identifiant → User File Directory (UFD)
- Répertoire par utilisateur: **User File Directory (UFD)**
 - 👉 Nom → FCB

STRUCTURE À DEUX NIVEAUX



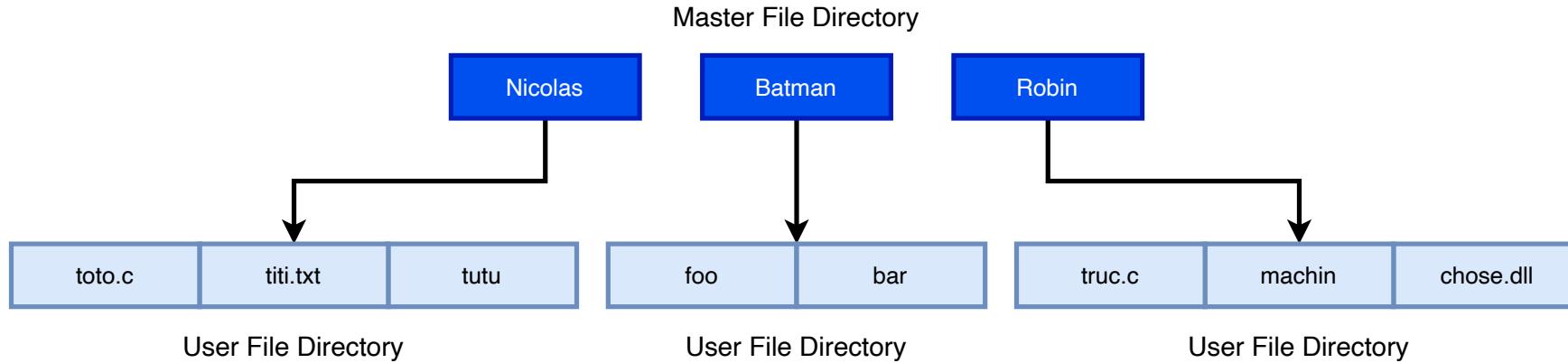
✓ User + Nom → FCB

STRUCTURE À DEUX NIVEAUX



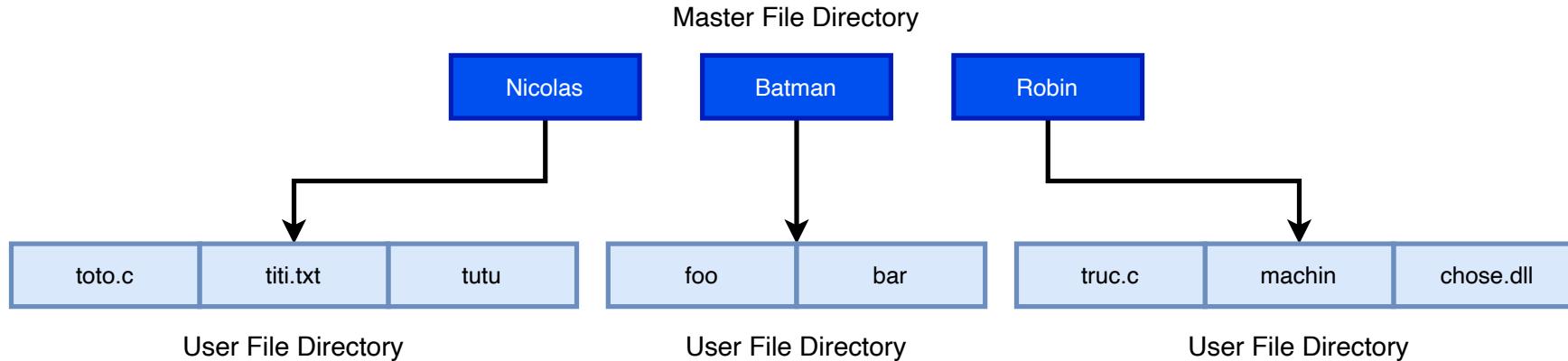
- ✓ User + Nom → FCB
- ✓ Pas beaucoup plus coûteux en taille

STRUCTURE À DEUX NIVEAUX



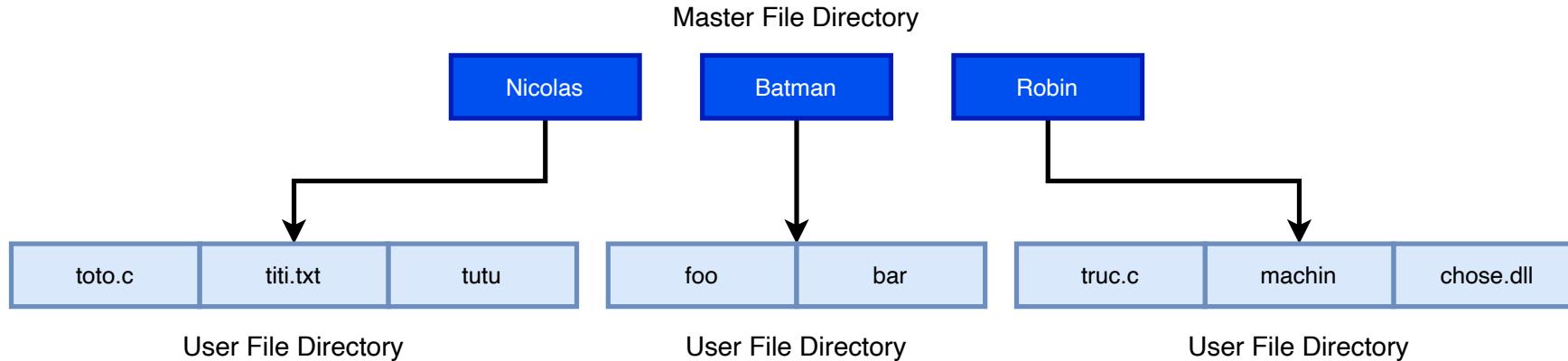
- ✓ User + Nom → FCB
- ✓ Pas beaucoup plus coûteux en taille
- ✓ Utilisateur : organiser les fichiers, unicité de nom, ...

STRUCTURE À DEUX NIVEAUX



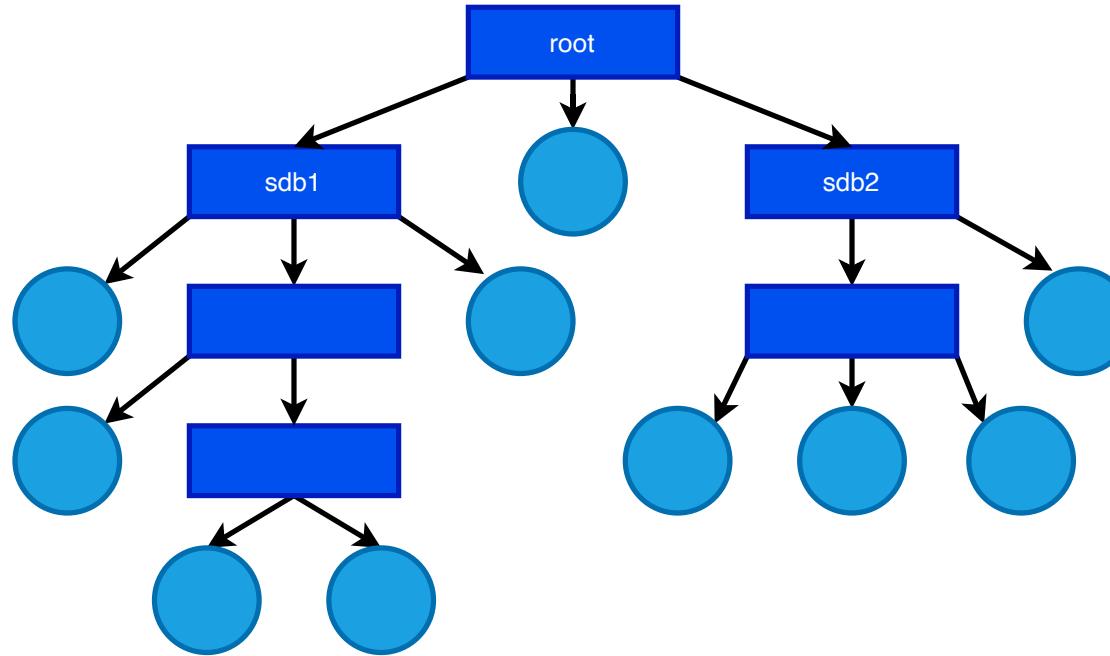
- ✓ User + Nom → FCB
- ✓ Pas beaucoup plus coûteux en taille
- ✓ Utilisateur : organiser les fichiers, unicité de nom, ...
- ✗ Taille des répertoires proportionnelle au nombre de fichiers

STRUCTURE À DEUX NIVEAUX

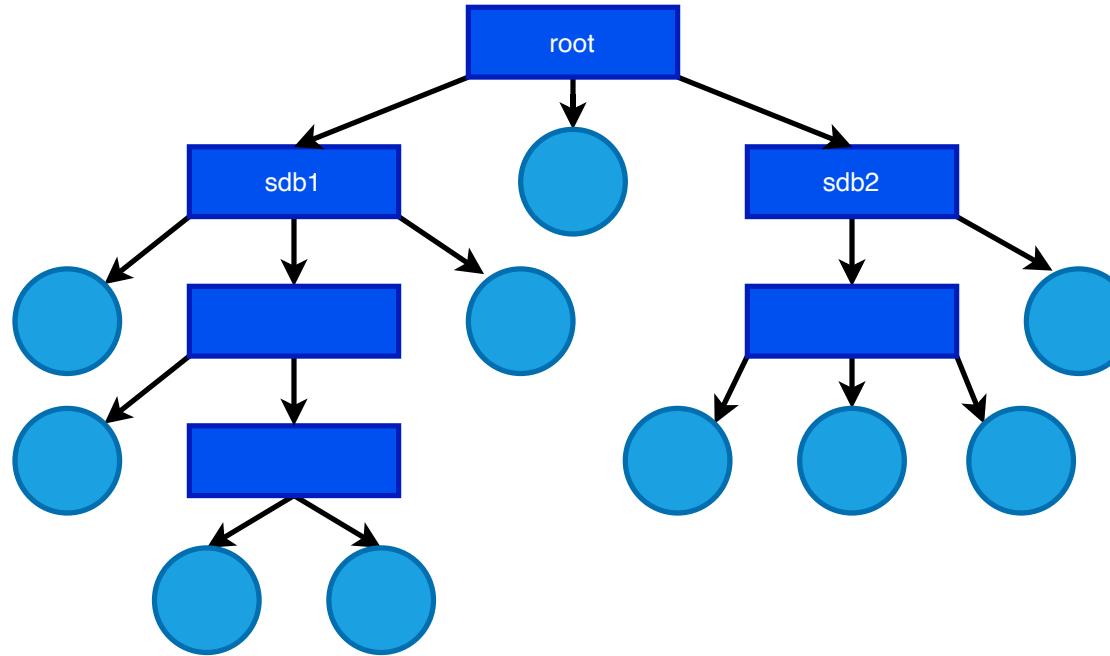


- ✓ User + Nom → FCB
- ✓ Pas beaucoup plus coûteux en taille
- ✓ Utilisateur : organiser les fichiers, unicité de nom, ...
- ✗ Taille des répertoires proportionnelle au nombre de fichiers
- ✗ Partage de fichiers

STRUCTURE ARBORESCENTE

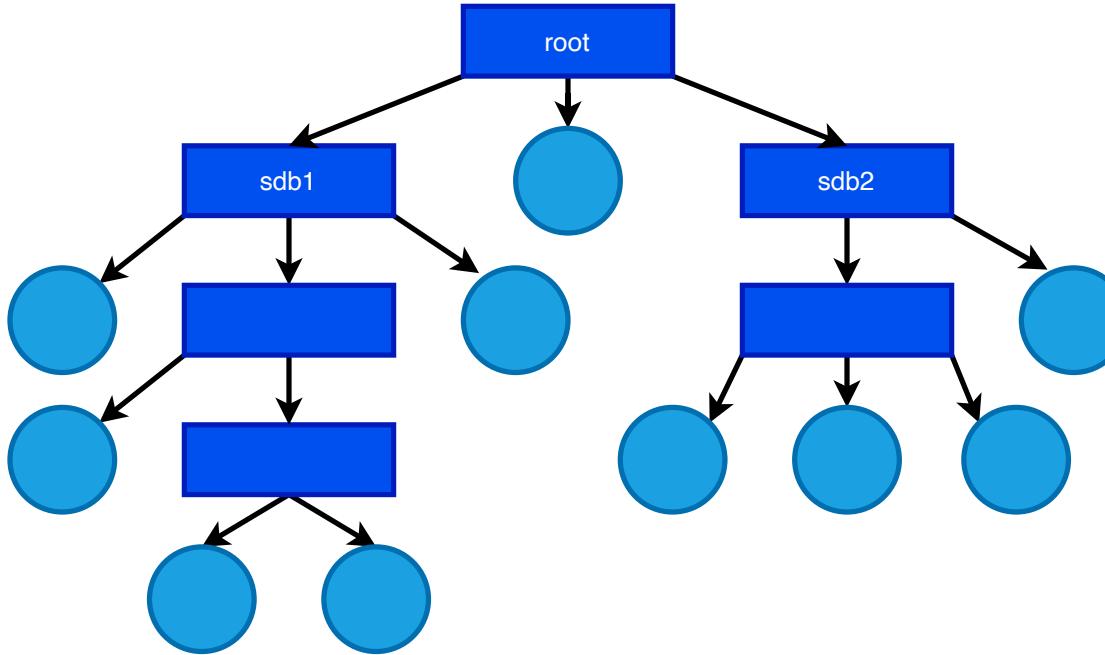


STRUCTURE ARBORESCENTE



Principe : généralisation de la structure à 2 niveaux

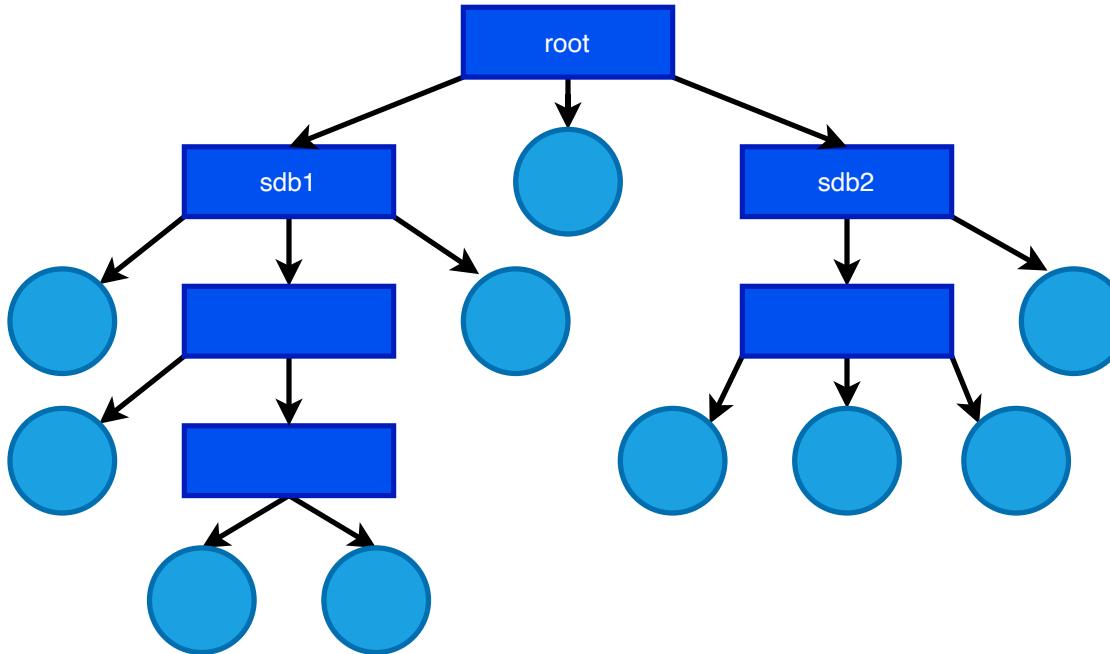
STRUCTURE ARBORESCENTE



Principe : généralisation de la structure à 2 niveaux

- **Répertoire racine (MFD)**

STRUCTURE ARBORESCENTE



Principe : généralisation de la structure à 2 niveaux

- **Répertoire racine (MFD)**
- **Sous-répertoires**, pouvant à leur tour jouer le rôle de **MFD**

IMPLEMENTATION

IMPLEMENTATION

Fichiers

IMPLEMENTATION

Fichiers

- Bit «*répertoire*» dans le FCB

IMPLEMENTATION

Fichiers

- Bit «*répertoire*» dans le **FCB**
- **Nom unique** = chemin depuis la racine (**chemin absolu**)

IMPLEMENTATION

Fichiers

- Bit «*répertoire*» dans le FCB
- Nom unique = chemin depuis la racine (**chemin absolu**)

OS

IMPLEMENTATION

Fichiers

- Bit «*répertoire*» dans le **FCB**
- **Nom unique** = chemin depuis la racine (**chemin absolu**)

OS

- Répertoire courant (par processus)

IMPLEMENTATION

Fichiers

- Bit «*répertoire*» dans le **FCB**
- **Nom unique** = chemin depuis la racine (**chemin absolu**)

OS

- Répertoire courant (par processus)
 - 👉 Recherche à partir du répertoire courant (**chemin relatif**)

IMPLEMENTATION

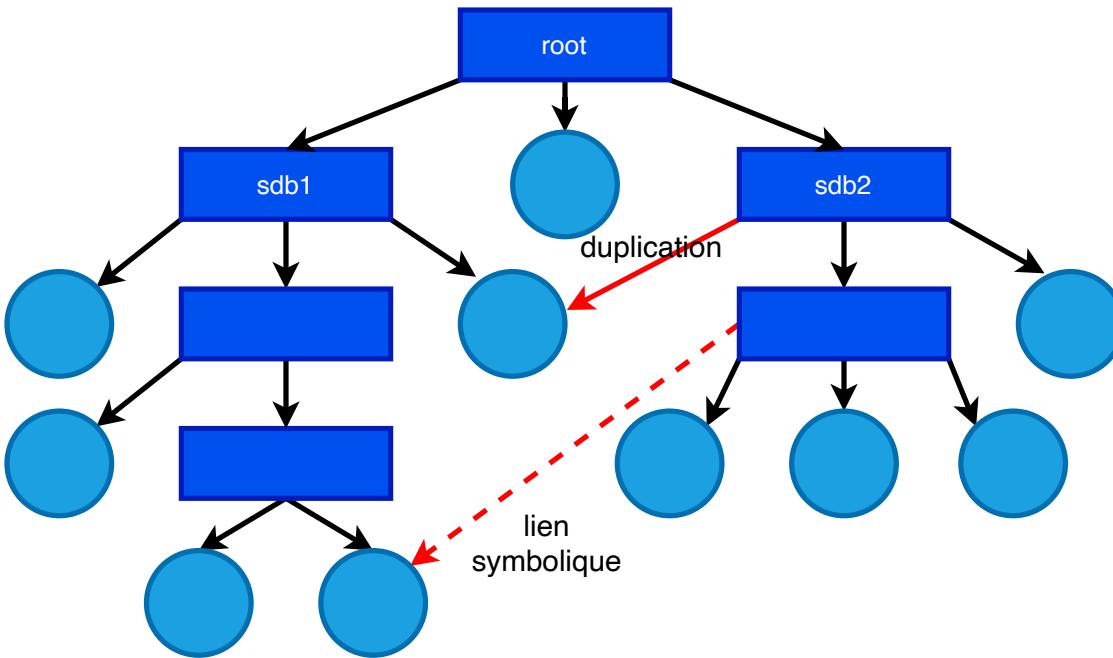
Fichiers

- Bit «*répertoire*» dans le **FCB**
- **Nom unique** = chemin depuis la racine (**chemin absolu**)

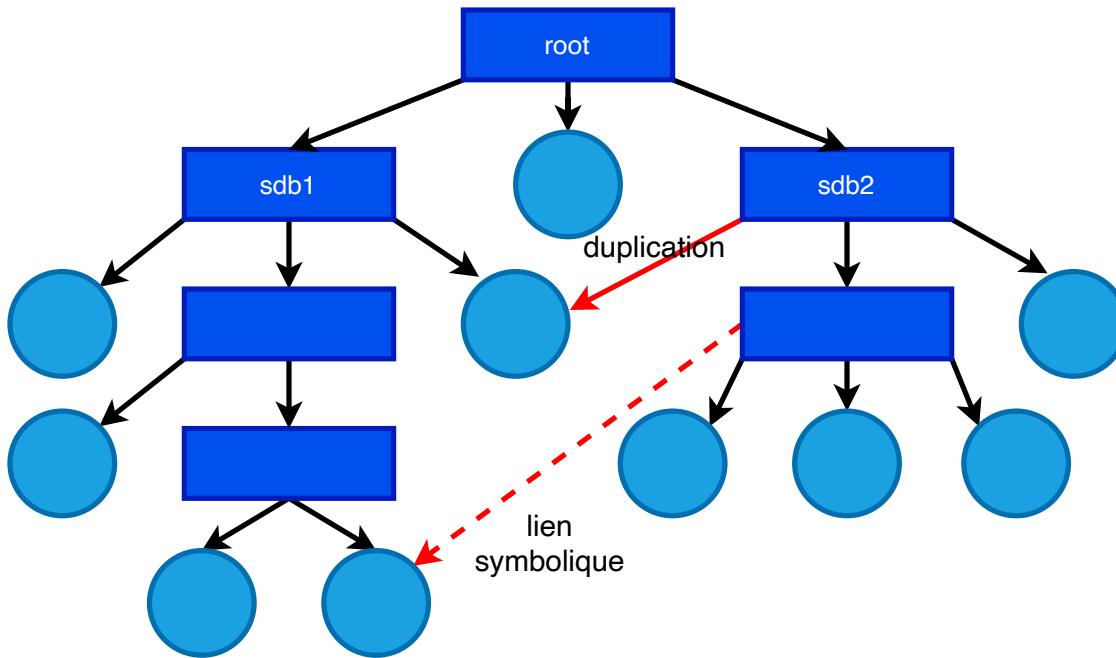
OS

- Répertoire courant (par processus)
 - 👉 Recherche à partir du répertoire courant (**chemin relatif**)
 - 👉 Recherche par défaut (**PATH**)

STRUCTURE EN GRAPHE

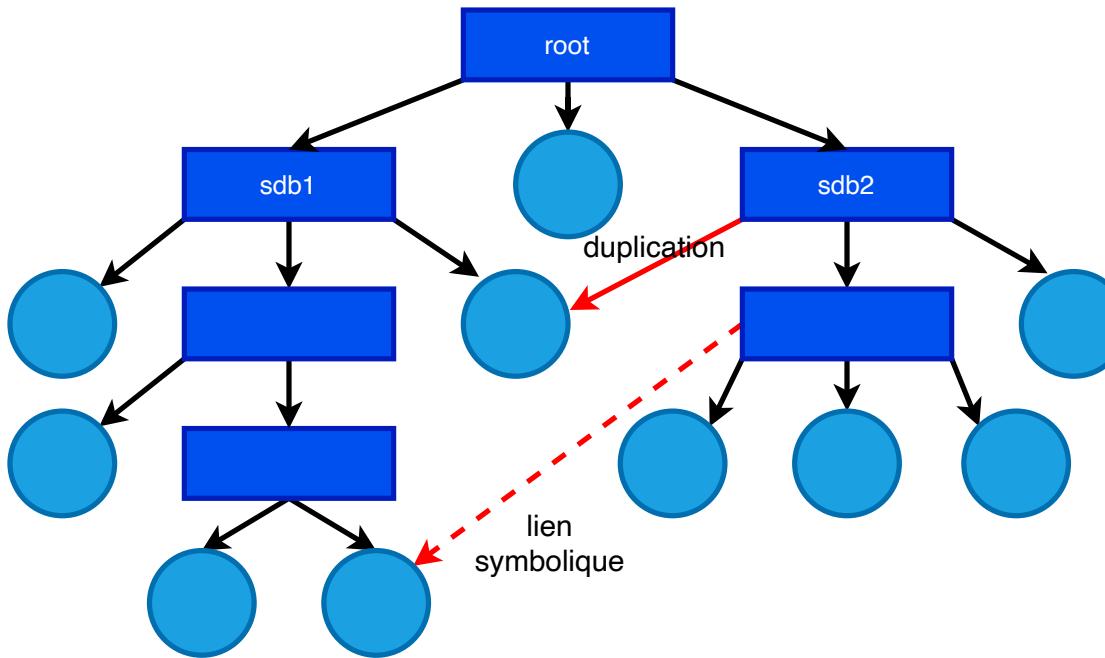


STRUCTURE EN GRAPHE



Principe : généralisation de l'arbre avec des **liens**

STRUCTURE EN GRAPHE



Principe : généralisation de l'arbre avec des **liens**
• Graphe **acyclique**

STRUCTURE EN GRAPHE : LIENS

STRUCTURE EN GRAPHE : LIENS

- Référencer un fichier décrit dans un autre répertoire

STRUCTURE EN GRAPHE : LIENS

- Référencer un fichier décrit dans un autre répertoire
👉 bit «*lien*» dans le répertoire + chemin absolu

STRUCTURE EN GRAPHE : LIENS

- Référencer un fichier décrit dans un autre répertoire
👉 bit «*lien*» dans le répertoire + chemin absolu
- Extension : duplication

STRUCTURE EN GRAPHE : LIENS

- **Référencer** un fichier décrit dans un autre répertoire
👉 bit «*lien*» dans le répertoire + chemin absolu
- **Extension : duplication**
👉 FCB recopié → copie et original indiscernables

STRUCTURE EN GRAPHE : LIENS

- **Référencer** un fichier décrit dans un autre répertoire
 - 👉 bit «*lien*» dans le répertoire + chemin absolu
- **Extension : duplication**
 - 👉 FCB recopié → copie et original indiscernables
 - 👉 Compteur de liens (savoir quand libérer l'espace sur le support physique)

OPÉRATIONS SUR UN FICHIER

OPÉRATIONS SUR UN FICHIER

- *Appels systèmes de base*

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture
 - **Écriture** : pointeur d'écriture

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture
 - **Écriture** : pointeur d'écriture
 - **Repositionnement** : déplacer un pointeur

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture
 - **Écriture** : pointeur d'écriture
 - **Repositionnement** : déplacer un pointeur
 - **Suppression** : retrait de l'entrée dans le répertoire

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture
 - **Écriture** : pointeur d'écriture
 - **Repositionnement** : déplacer un pointeur
 - **Suppression** : retrait de l'entrée dans le répertoire
 - **Troncature** : vider mais garder l'entrée

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture
 - **Écriture** : pointeur d'écriture
 - **Repositionnement** : déplacer un pointeur
 - **Suppression** : retrait de l'entrée dans le répertoire
 - **Troncature** : vider mais garder l'entrée
- **Opérations composées**

OPÉRATIONS SUR UN FICHIER

- **Appels systèmes de base**
 - **Création** : allocation espace + entrée **répertoire**
 - **Lecture** : pointeur de lecture
 - **Écriture** : pointeur d'écriture
 - **Repositionnement** : déplacer un pointeur
 - **Suppression** : retrait de l'entrée dans le répertoire
 - **Troncature** : vider mais garder l'entrée
- **Opérations composées**
 - **Ex:** copie, renommage → effectuées à partir des appels systèmes de base

OUVERTURE DE FICHIER

OUVERTURE DE FICHIER

Problème

OUVERTURE DE FICHIER

Problème

- Nécessité d'accéder au **FCB** à chaque opération sur le fichier.

OUVERTURE DE FICHIER

Problème

- Nécessité d'accéder au **FCB** à chaque opération sur le fichier.
- Le **FCB** est stocké dans le répertoire du périphérique.

OUVERTURE DE FICHIER

Problème

- Nécessité d'accéder au **FCB** à chaque opération sur le fichier.
 - Le **FCB** est stocké dans le répertoire du périphérique.
- 👉 Très coûteux en accès disque (donc en temps)!

OUVERTURE DE FICHIER

Problème

- Nécessité d'accéder au **FCB** à chaque opération sur le fichier.
 - Le **FCB** est stocké dans le répertoire du périphérique.
- 👉 Très coûteux en accès disque (donc en temps)!

Solution

- L'appel système **open** permet de charger le **FCB** en mémoire.

OUVERTURE DE FICHIER

Problème

- Nécessité d'accéder au **FCB** à chaque opération sur le fichier.
 - Le **FCB** est stocké dans le répertoire du périphérique.
- 👉 Très coûteux en accès disque (donc en temps)!

Solution

- L'appel système **open** permet de charger le **FCB** en mémoire.
- L'OS impose que tout accès à un fichier soit précédé d'un **open**.

TABLE DES FICHIERS OUVERTS

TABLE DES FICHIERS OUVERTS

Stockage des FCB en RAM

TABLE DES FICHIERS OUVERTS

Stockage des FCB en RAM

- La **table des fichiers ouverts** de l'OS contient l'ensemble des **FCB** des fichiers ouverts.

TABLE DES FICHIERS OUVERTS

Stockage des FCB en RAM

- La **table des fichiers ouverts** de l'OS contient l'ensemble des **FCB** des fichiers ouverts.
 - Ouverture → chargement du **FCB** + ajout dans la table

TABLE DES FICHIERS OUVERTS

Stockage des FCB en RAM

- La **table des fichiers ouverts** de l'OS contient l'ensemble des **FCB** des fichiers ouverts.
 - Ouverture → chargement du **FCB** + ajout dans la table
 - Fermeture → retrait de la table

TABLE DES FICHIERS OUVERTS

Stockage des FCB en RAM

- La **table des fichiers ouverts** de l'OS contient l'ensemble des **FCB** des fichiers ouverts.
 - Ouverture → chargement du **FCB** + ajout dans la table
 - Fermeture → retrait de la table
 - Pas d'impact sur le fichier!

TABLE DES FICHIERS OUVERTS

Gestion par l'OS

TABLE DES FICHIERS OUVERTS

Gestion par l'OS

- **Implicite** : `open` au premier accès

TABLE DES FICHIERS OUVERTS

Gestion par l'OS

- **Implicite** : `open` au premier accès
- **Explicite** : `exception` si le fichier n'a pas été ouvert avant

TABLE DES FICHIERS OUVERTS

Gestion par l'OS

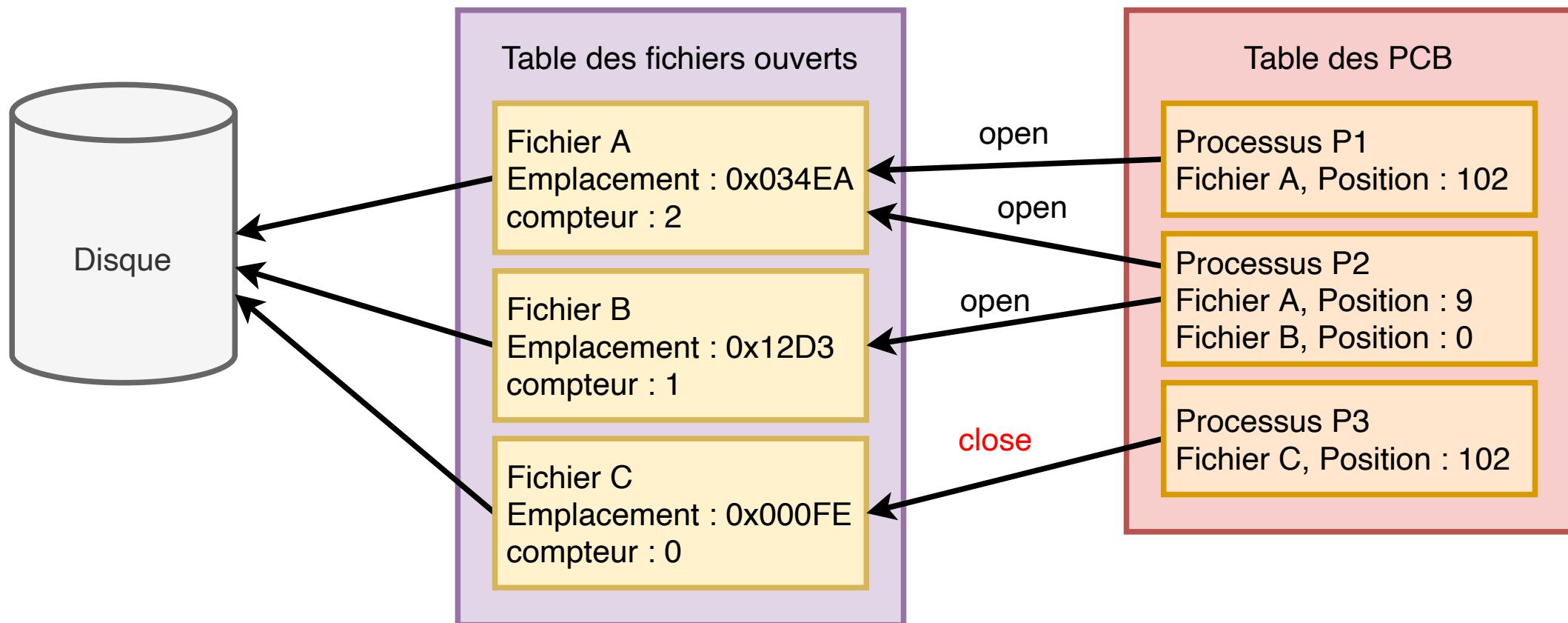
- **Implicite** : `open` au premier accès
- **Explicite** : `exception` si le fichier n'a pas été ouvert avant
- Une table de fichiers ouverts **globale** avec compteurs
+ une table **par processus** → fermeture à la terminaison

TABLE DES FICHIERS OUVERTS

Gestion par l'OS

- **Implicite** : `open` au premier accès
- **Explicite** : `exception` si le fichier n'a pas été ouvert avant
- Une table de fichiers ouverts **globale** avec compteurs
 - + une table **par processus** → fermeture à la terminaison
- Possibilité d'interdire l'accès aux autres processus

EXEMPLE



PROTECTIONS

PROTECTIONS

- La protection permet de spécifier pour chaque fichier la liste des sujets autorisés à effectuer un type d'accès :
👉 ACL (Access control list).

PROTECTIONS

- La protection permet de spécifier pour chaque fichier la liste des sujets autorisés à effectuer un type d'accès :
👉 ACL (Access control list).
- Dans les systèmes POSIX on distingue :

PROTECTIONS

- La protection permet de spécifier pour chaque fichier la liste des sujets autorisés à effectuer un type d'accès :
👉 ACL (Access control list).
- Dans les systèmes POSIX on distingue :
 - 3 modes (lecture, écriture, exécution)

PROTECTIONS

- La protection permet de spécifier pour chaque fichier la liste des sujets autorisés à effectuer un type d'accès :
👉 ACL (Access control list).
- Dans les systèmes POSIX on distingue :
 - 3 modes (lecture, écriture, exécution)
 - 3 catégories de sujets (le propriétaire, son groupe et le reste des sujets)

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier			
Répertoire			

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier	ouverture en mode lecture		
Répertoire			

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier	ouverture en mode lecture	ouverture en mode écriture	
Répertoire			

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier	ouverture en mode lecture	ouverture en mode écriture	exécution du fichier
Répertoire			

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier	ouverture en mode lecture	ouverture en mode écriture	exécution du fichier
Répertoire	lister les fichiers contenus		

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier	ouverture en mode lecture	ouverture en mode écriture	exécution du fichier
Répertoire	lister les fichiers contenus	créer, renommer supprimer un fichier	

PROTECTIONS

	Lecture (R)	Écriture (W)	Exécution (X)
Fichier	ouverture en mode lecture	ouverture en mode écriture	exécution du fichier
Répertoire	lister les fichiers contenus	créer, renommer supprimer un fichier	entrer dans le répertoire et accéder à son contenu

PARTITIONS, MONTAGE ET VFS

PARTITIONS, MONTAGE ET VFS

- Les supports physiques sont partitionnés (découpés) en sous-ensembles logiques sur lesquels un FS (File System) est installé

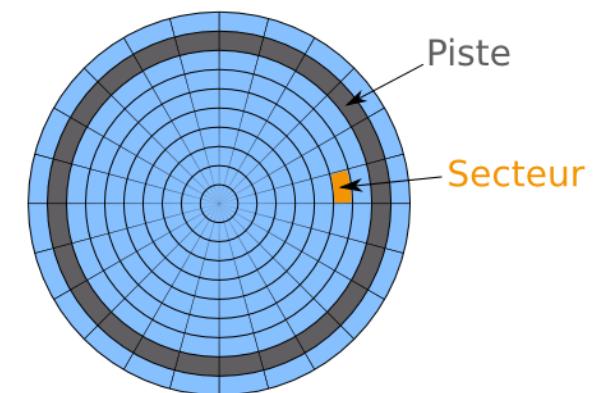
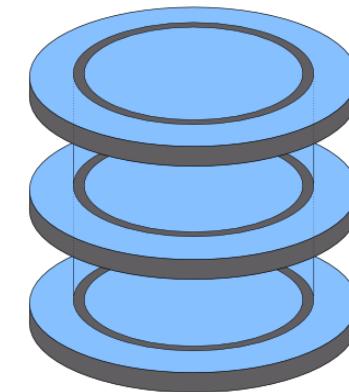
PARTITIONS, MONTAGE ET VFS

- Les supports physiques sont partitionnés (découpés) en sous-ensembles logiques sur lesquels un FS (File System) est installé
- L'OS rend disponible les divers FS déclarés dans la table de montage (on parle aussi de Volumes)

PARTITIONS, MONTAGE ET VFS

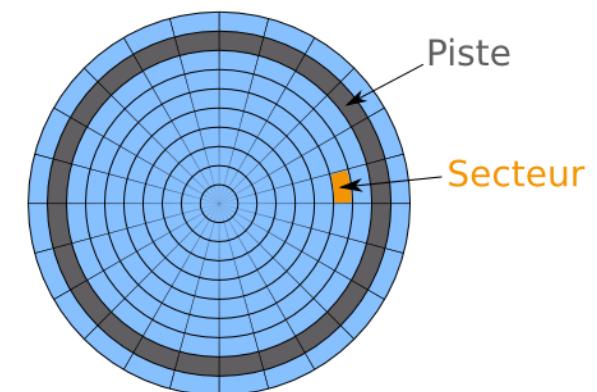
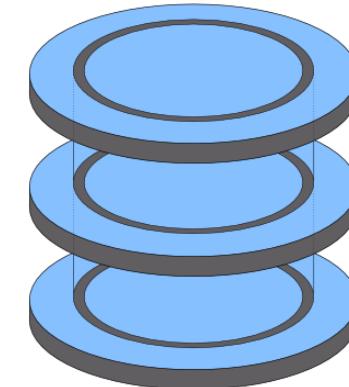
- Les supports physiques sont partitionnés (découpés) en sous-ensembles logiques sur lesquels un FS (File System) est installé
- L'OS rend disponible les divers FS déclarés dans la table de montage (on parle aussi de Volumes)
- L'OS présente une vue unifiée des différents Volumes (FS) disponibles (Virtual File System - VFS)

LA VUE PHYSIQUE



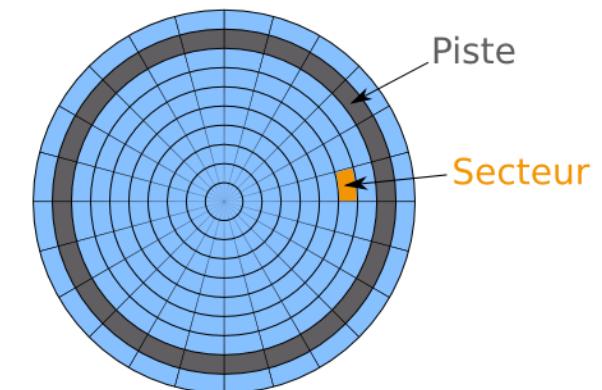
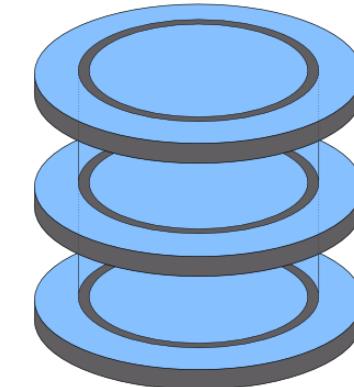
LA VUE PHYSIQUE

- Les supports de stockage sont **décomposés en blocs** (les éléments atomiques du **FS**).



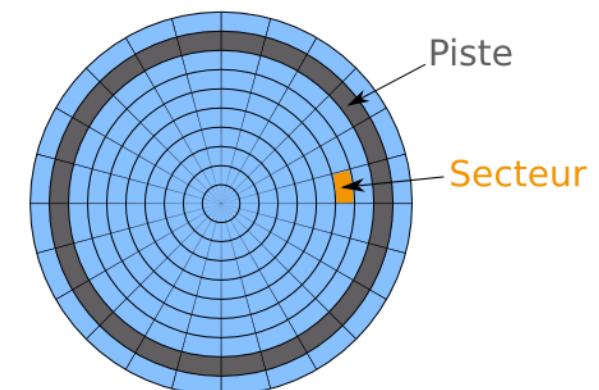
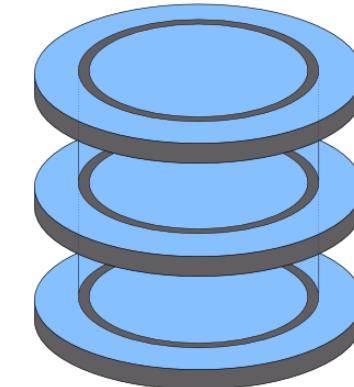
LA VUE PHYSIQUE

- Les supports de stockage sont **décomposés en blocs** (les éléments atomiques du **FS**).
- Selon les technologies, la **création des blocs** et **émule** toujours celle des **disques durs**:



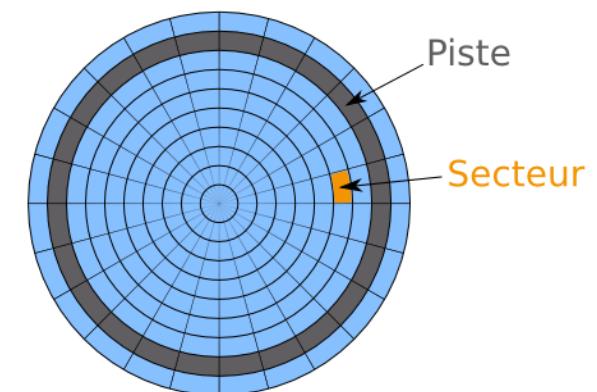
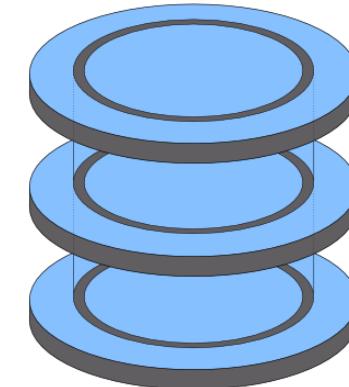
LA VUE PHYSIQUE

- Les supports de stockage sont **décomposés en blocs** (les éléments atomiques du **FS**).
- Selon les technologies, la **création des blocs** et **émule** toujours celle des **disques durs**:
- **Le formatage de bas niveau :**



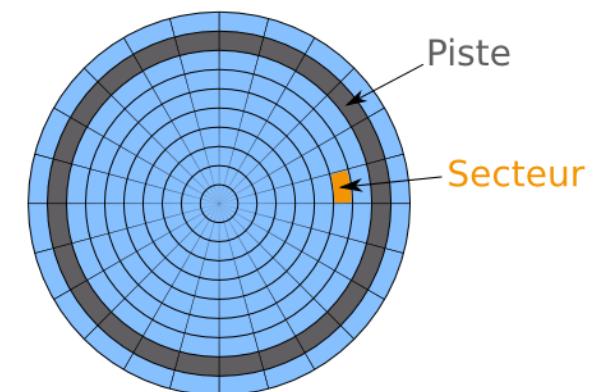
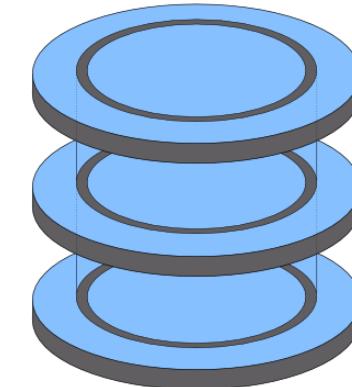
LA VUE PHYSIQUE

- Les supports de stockage sont **décomposés en blocs** (les éléments atomiques du **FS**).
- Selon les technologies, la **création des blocs** et **émule** toujours celle des **disques durs**:
- **Le formatage de bas niveau :**
 1. décomposer en **secteurs** (identifiés par cylindre, tête de lecture et secteur de la piste)



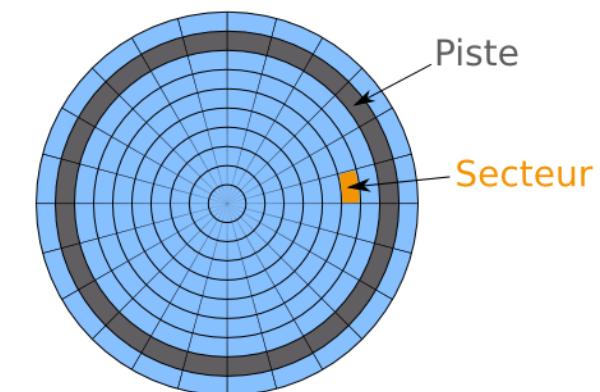
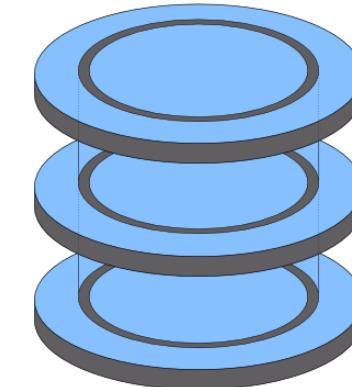
LA VUE PHYSIQUE

- Les supports de stockage sont **décomposés en blocs** (les éléments atomiques du **FS**).
- Selon les technologies, la **création des blocs** et **émule** toujours celle des **disques durs**:
- **Le formatage de bas niveau :**
 1. décomposer en **secteurs** (identifiés par cylindre, tête de lecture et secteur de la piste)
 2. puis les regrouper en **blocs**.



LA VUE PHYSIQUE

- Les supports de stockage sont **décomposés en blocs** (les éléments atomiques du **FS**).
- Selon les technologies, la **création des blocs** et **émule** toujours celle des **disques durs**:
- **Le formatage de bas niveau :**
 1. décomposer en **secteurs** (identifiés par cylindre, tête de lecture et secteur de la piste)
 2. puis les regrouper en **blocs**.



Le découpage en blocs génère de **la fragmentation interne** (l'espace inutilisé du dernier bloc).

SYSTÈME DE FICHIER

SYSTÈME DE FICHIER

Questions !

- Comment stocker les données et les programmes sur le disque

SYSTÈME DE FICHIER

Questions !

- Comment stocker les données et les programmes sur le disque
👉 Comment les **organiser** ?

SYSTÈME DE FICHIER

Questions !

- Comment stocker les données et les programmes sur le disque
 - 👉 Comment les **organiser** ?
 - 👉 Comment y **accéder** ?

SYSTÈME DE FICHIER

Questions !

- Comment stocker les données et les programmes sur le disque
 - 👉 Comment les **organiser** ?
 - 👉 Comment y **accéder** ?

Réponse !

- Définir une **norme** de gestion (**Ex: FAT, NTFS, EXT4FS, NFS ...**)

SYSTÈME DE FICHIER

Questions !

- Comment stocker les données et les programmes sur le disque
 - 👉 Comment les **organiser** ?
 - 👉 Comment y **accéder** ?

Réponse !

- Définir une **norme** de gestion (**Ex: FAT, NTFS, EXT4FS, NFS ...**)
 - 👉 Organisation de l'ensemble des données et des périphériques

SYSTÈME DE FICHIER

Questions !

- Comment stocker les données et les programmes sur le disque
 - 👉 Comment les **organiser** ?
 - 👉 Comment y **accéder** ?

Réponse !

- Définir une **norme** de gestion (**Ex: FAT, NTFS, EXT4FS, NFS ...**)
 - 👉 Organisation de l'ensemble des données et des périphériques
 - 👉 **Ex: Linux** → chaque **périphérique** est représenté par un **fichier**

STRUCTURE D'UN SYSTÈME DE FICHIERS

STRUCTURE D'UN SYSTÈME DE FICHIERS

- *Système logique*

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection
- **Système physique**

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection
- **Système physique**
 - Fichiers → ensemble de blocs logiques

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection
- **Système physique**
 - Fichiers → ensemble de blocs logiques
 - Bloc logique → blocs physique

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection
- **Système physique**
 - Fichiers → ensemble de blocs logiques
 - Bloc logique → blocs physique
 - Identification des blocs physiques selon support

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection
- **Système physique**
 - Fichiers → ensemble de blocs logiques
 - Bloc logique → blocs physique
 - Identification des blocs physiques selon support
- **Lien : pilote de périphérique**

STRUCTURE D'UN SYSTÈME DE FICHIERS

- **Système logique**
 - Structure de répertoires
 - FCB + gestion de la protection
- **Système physique**
 - Fichiers → ensemble de blocs logiques
 - Bloc logique → blocs physique
 - Identification des blocs physiques selon support
- **Lien : pilote de périphérique**
 - Appel système (ex: chargement bloc 456) → instruction matériel

MONTAGE DE RÉPERTOIRE

MONTAGE DE RÉPERTOIRE

- *Le système de fichiers*

MONTAGE DE RÉPERTOIRE

- **Le système de fichiers**
 - associe des noms à des blocs logiques (**fichiers**)

MONTAGE DE RÉPERTOIRE

- **Le système de fichiers**
 - associe des noms à des blocs logiques (**fichiers**)
 - associe des noms à des **répertoires** venus du disque!

MONTAGE DE RÉPERTOIRE

- **Le système de fichiers**
 - associe des noms à des blocs logiques (**fichiers**)
 - associe des noms à des **répertoires** venus du disque!
- **Montage de répertoire**
 - consiste à positionner un répertoire dans le système de fichier

MONTAGE DE RÉPERTOIRE

Principe

MONTAGE DE RÉPERTOIRE

Principe

- Cas 1 : chargement du **FCB** par l'**OS** (disque → **RAM**)

MONTAGE DE RÉPERTOIRE

Principe

- Cas 1 : chargement du **FCB** par l'**OS** (disque → **RAM**)
👉 attribué à l'utilisateur du processus

MONTAGE DE RÉPERTOIRE

Principe

- Cas 1 : chargement du **FCB** par l'**OS** (disque → **RAM**)
👉 attribué à l'utilisateur du processus
- Cas 2 : association dans le **MFD** au niveau **logique**

MONTAGE DE RÉPERTOIRE

Principe

- **Cas 1** : chargement du **FCB** par l'**OS** (disque → **RAM**)
👉 attribué à l'utilisateur du processus
- **Cas 2** : association dans le **MFD** au niveau **logique**
👉 les fichiers deviennent accessibles

IMPLEMENTATION

Sur le disque

IMPLÉMENTATION

Sur le disque

- **Bloc de démarrage (Boot Control Block)**
👉 chargement de l'OS depuis une partition

IMPLÉMENTATION

Sur le disque

- **Bloc de démarrage (Boot Control Block)**
 - 👉 chargement de l'OS depuis une partition
- **Bloc de contrôle de partition (Volume Control Block)**
 - 👉 nombre de blocs, leur taille,
 - 👉 les blocs libres, les structures de descriptions de fichiers libres, ...

IMPLÉMENTATION

Sur le disque

- **Bloc de démarrage (Boot Control Block)**
 - 👉 chargement de l'OS depuis une partition
- **Bloc de contrôle de partition (Volume Control Block)**
 - 👉 nombre de blocs, leur taille,
 - 👉 les blocs libres, les structures de descriptions de fichiers libres, ...
- **Bloc de contrôle de répertoire ou de fichier (FCB)**

IMPLEMENTATION

Au niveau de l'OS

IMPLÉMENTATION

Au niveau de l'OS

- Table des partitions/répertoires montés

IMPLÉMENTATION

Au niveau de l'OS

- Table des partitions/répertoires montés
- Cache des répertoires

IMPLÉMENTATION

Au niveau de l'OS

- Table des partitions/répertoires montés
- Cache des répertoires
- Table des fichiers ouverts (copie des **FCB**)

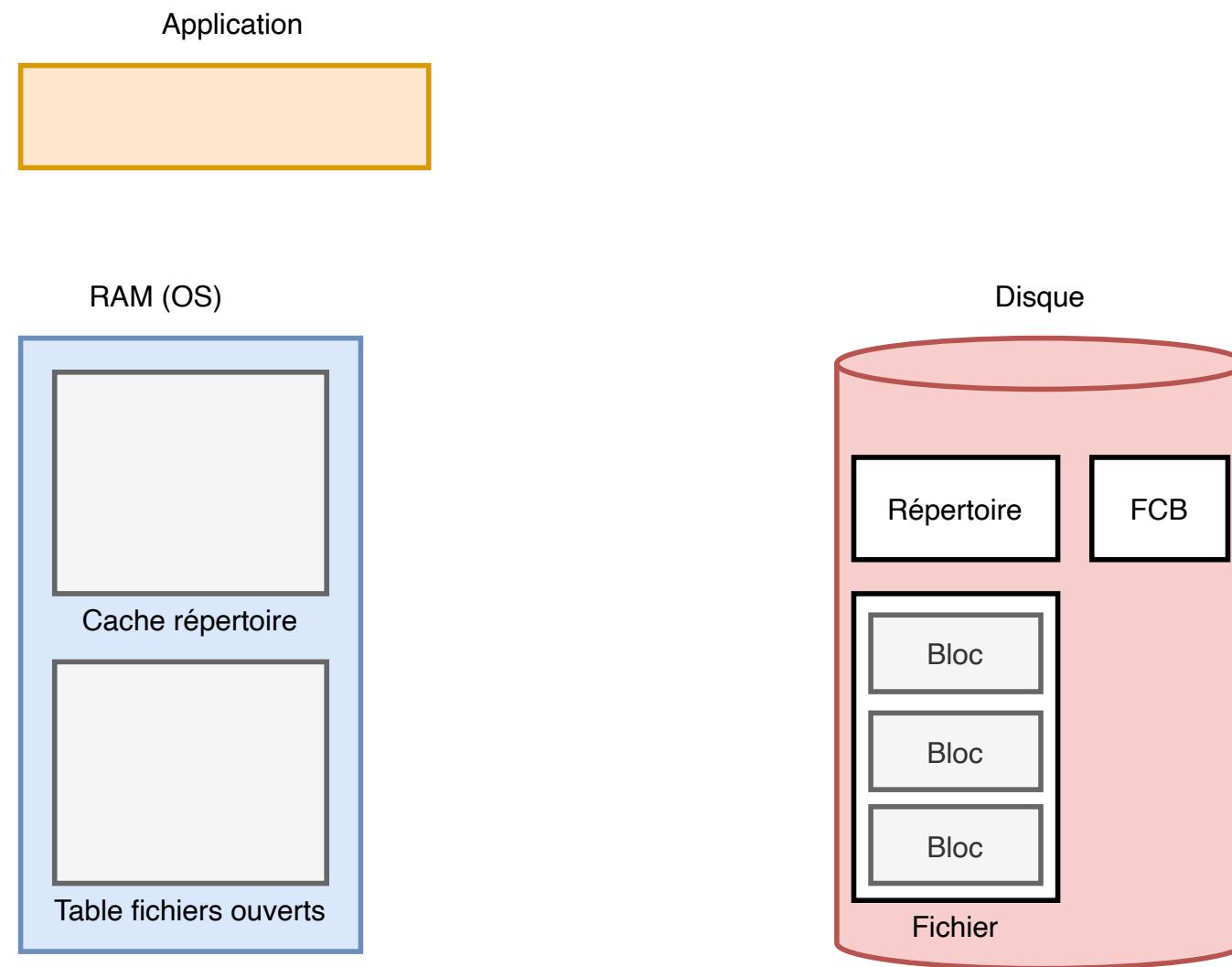
IMPLÉMENTATION

Au niveau de l'OS

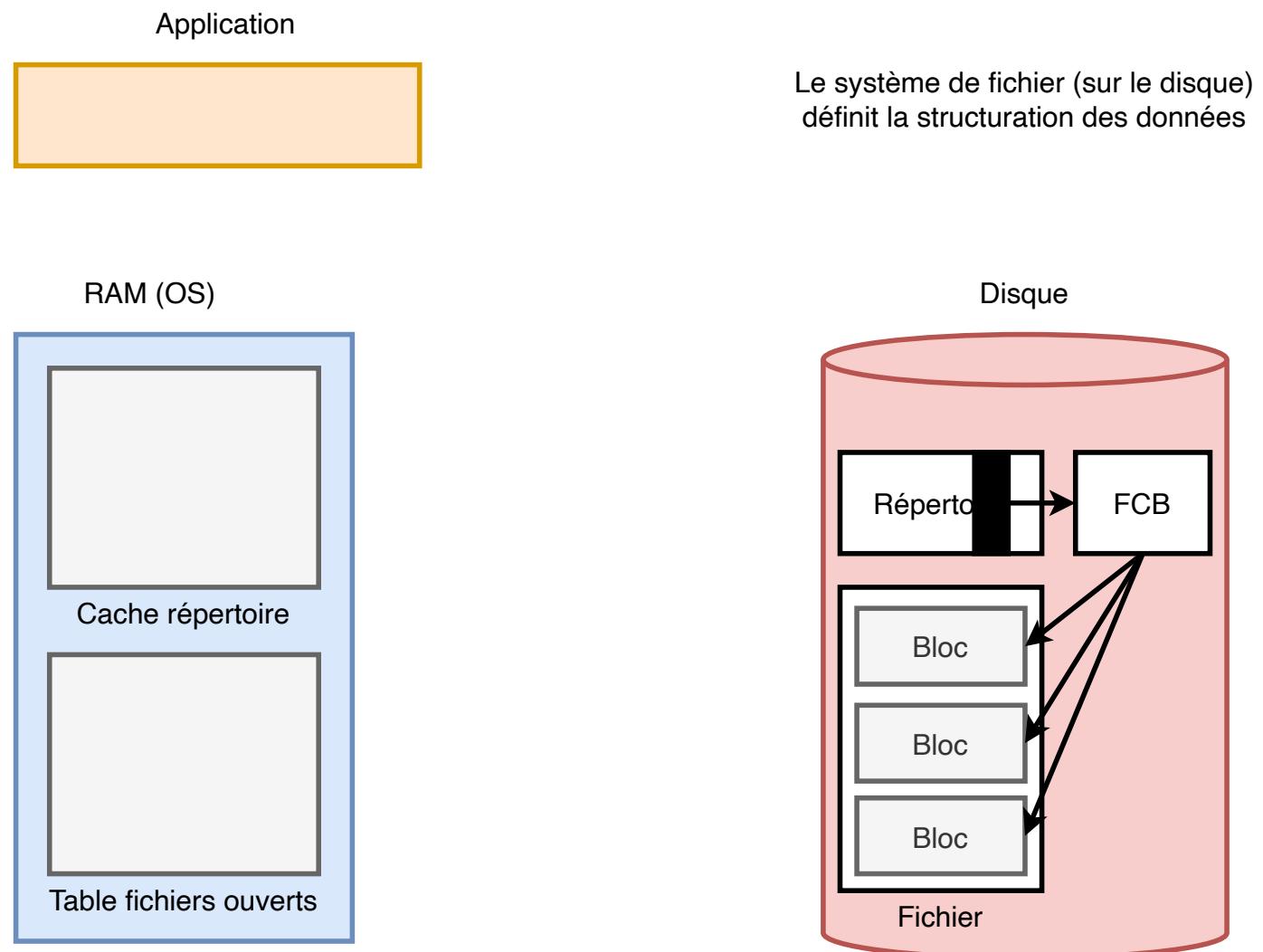
- Table des partitions/répertoires montés
- Cache des répertoires
- Table des fichiers ouverts (copie des **FCB**)
- Blocs logiques

ACCÈS À UN FICHIER

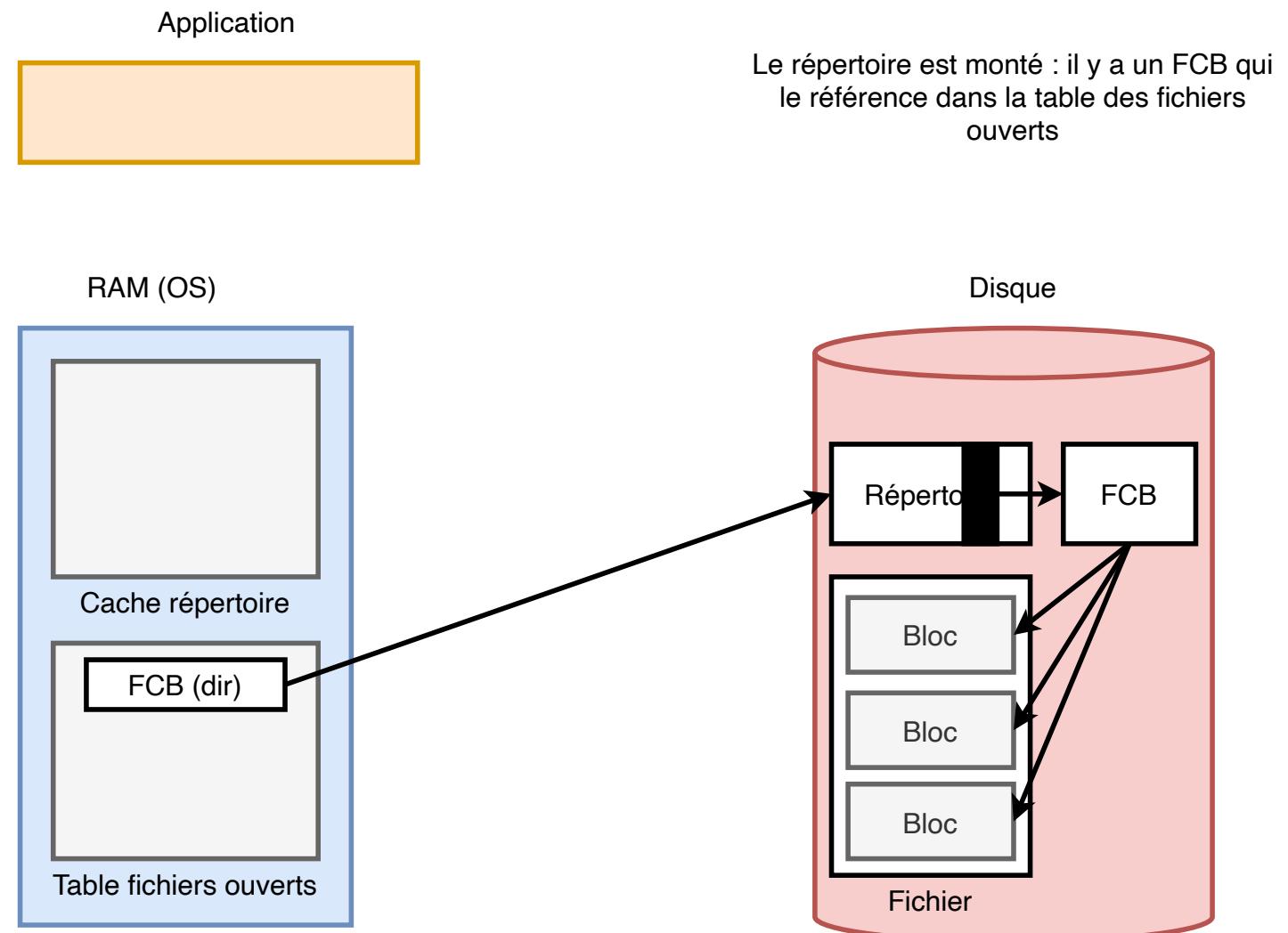
ACCÈS À UN FICHIER



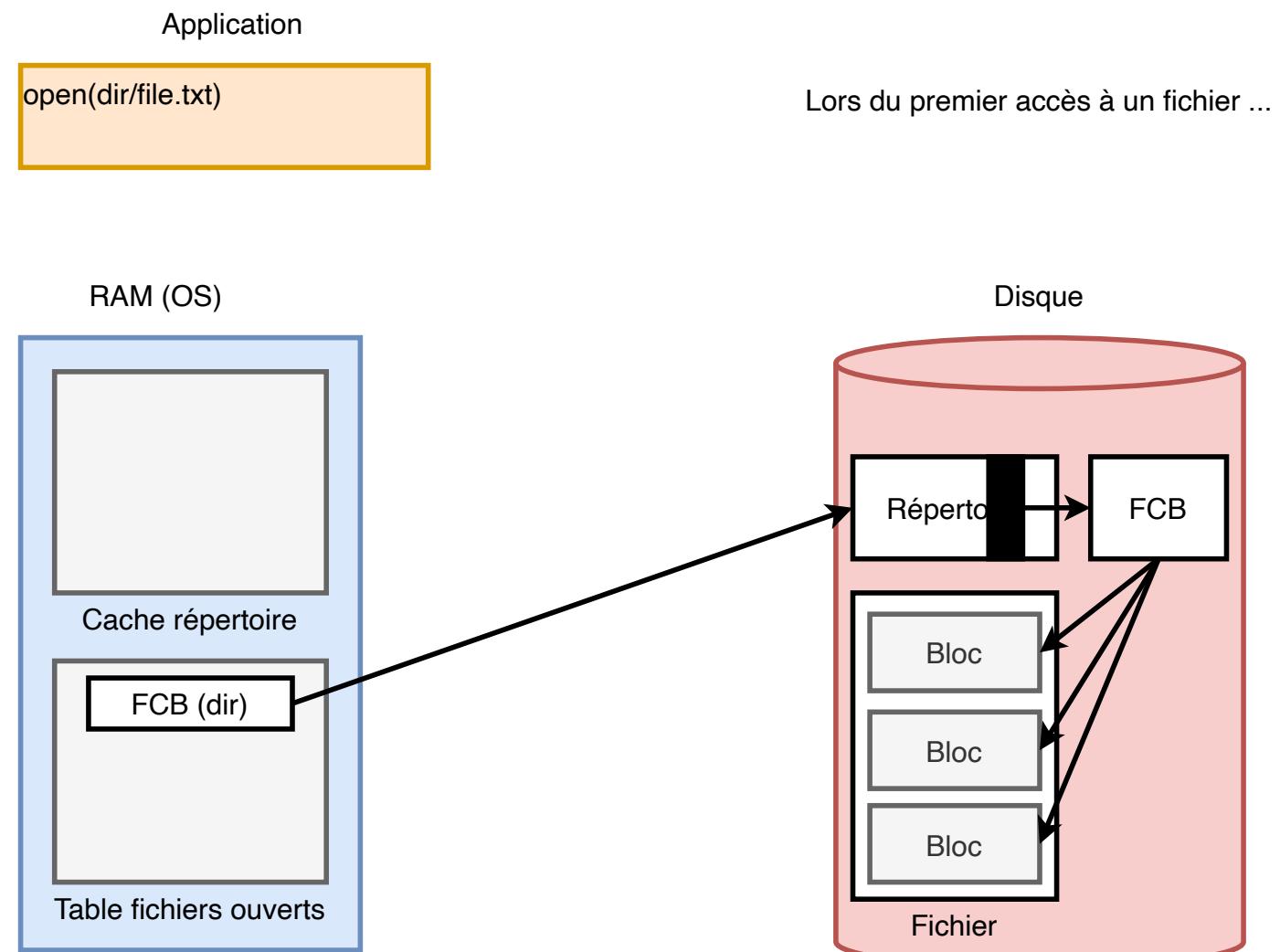
ACCÈS À UN FICHIER



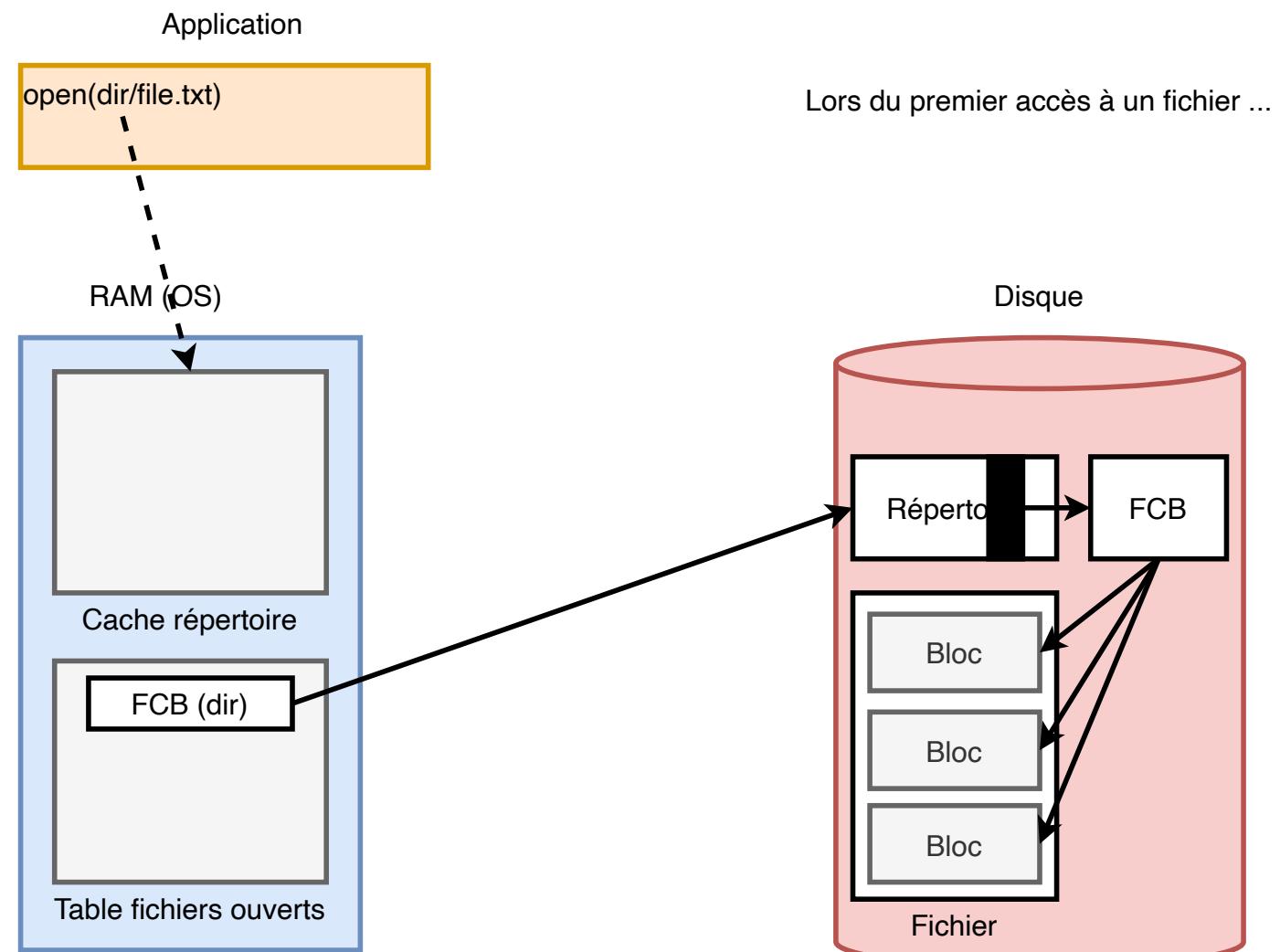
ACCÈS À UN FICHIER



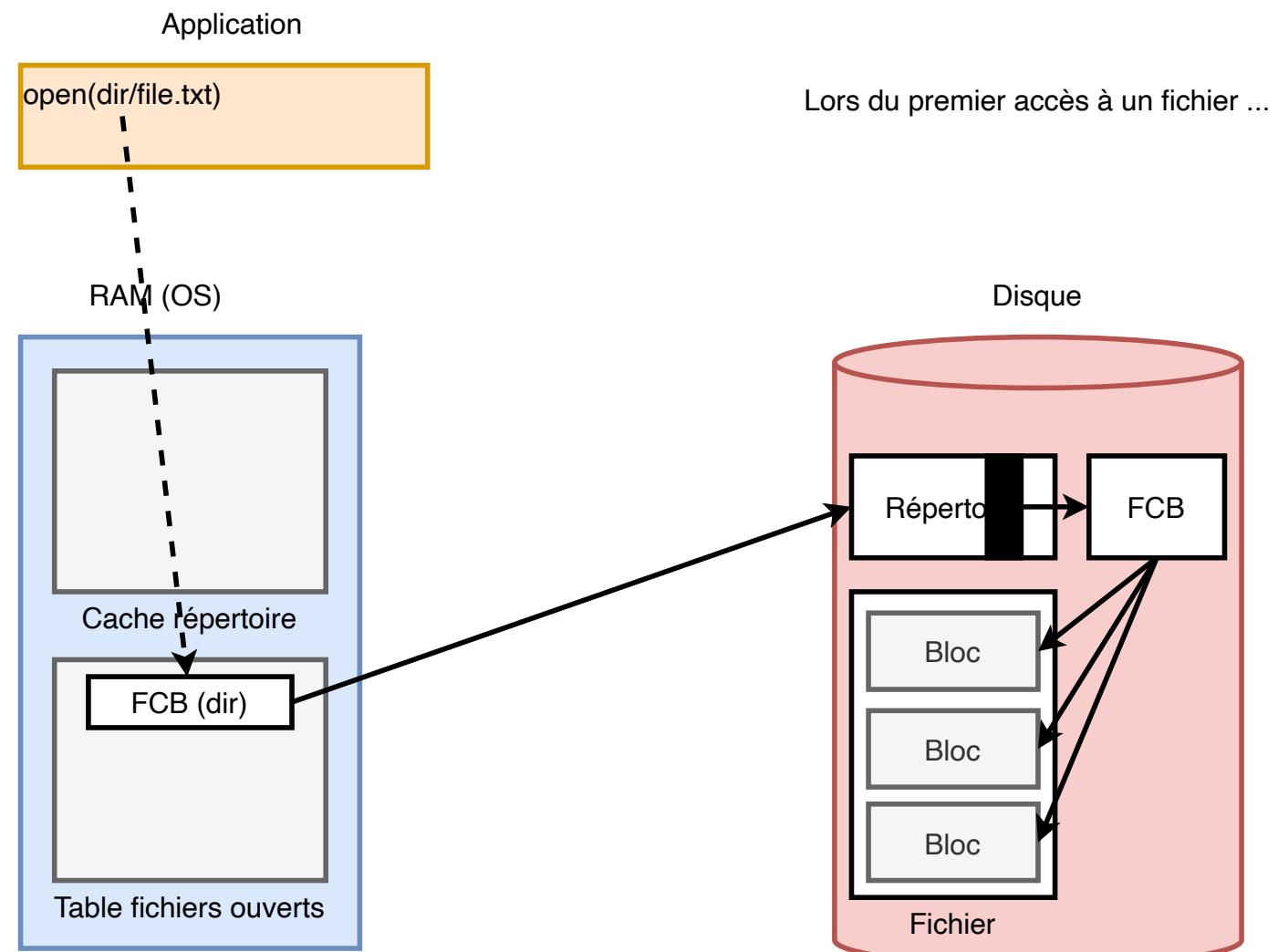
ACCÈS À UN FICHIER



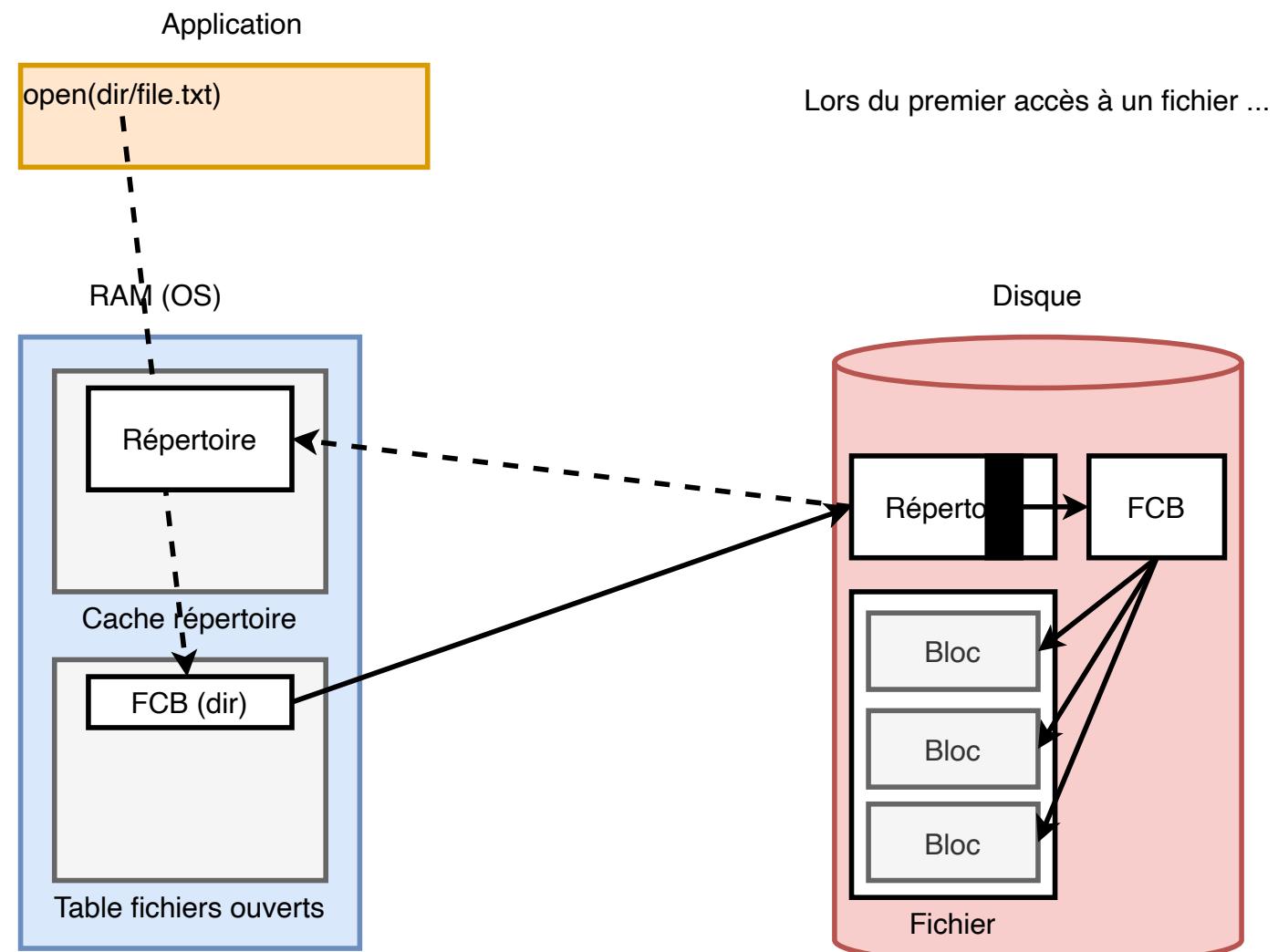
ACCÈS À UN FICHIER



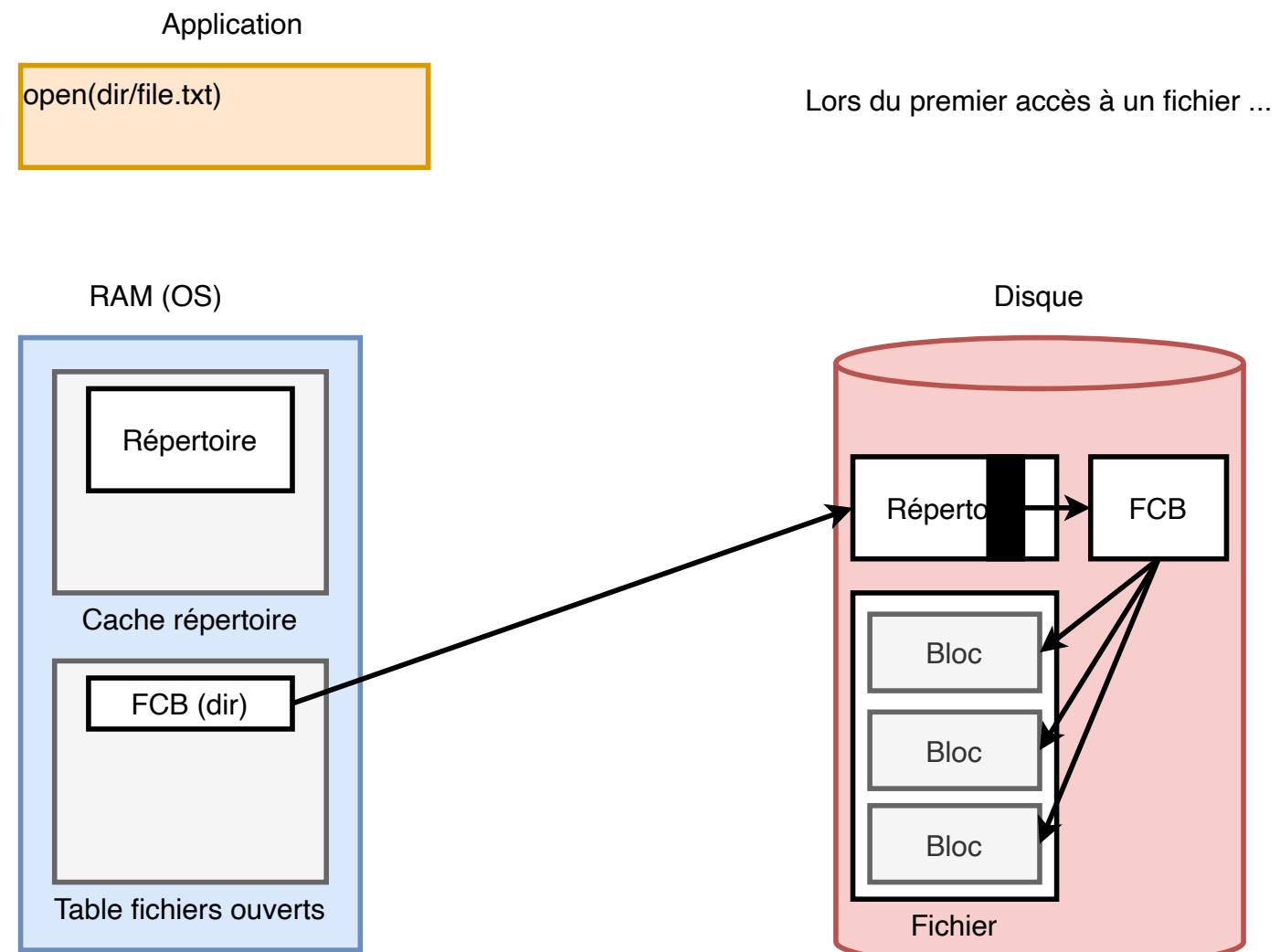
ACCÈS À UN FICHIER



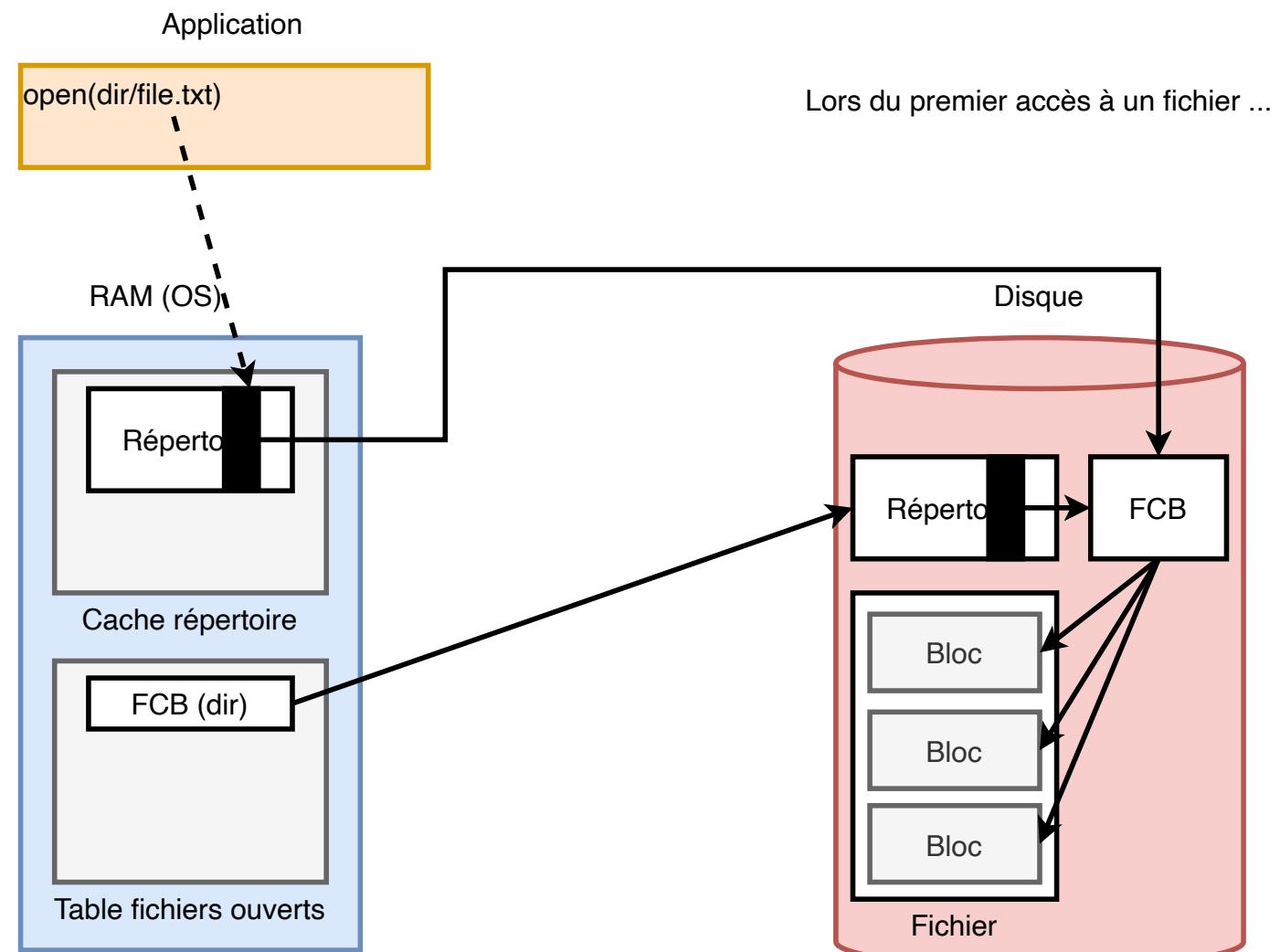
ACCÈS À UN FICHIER



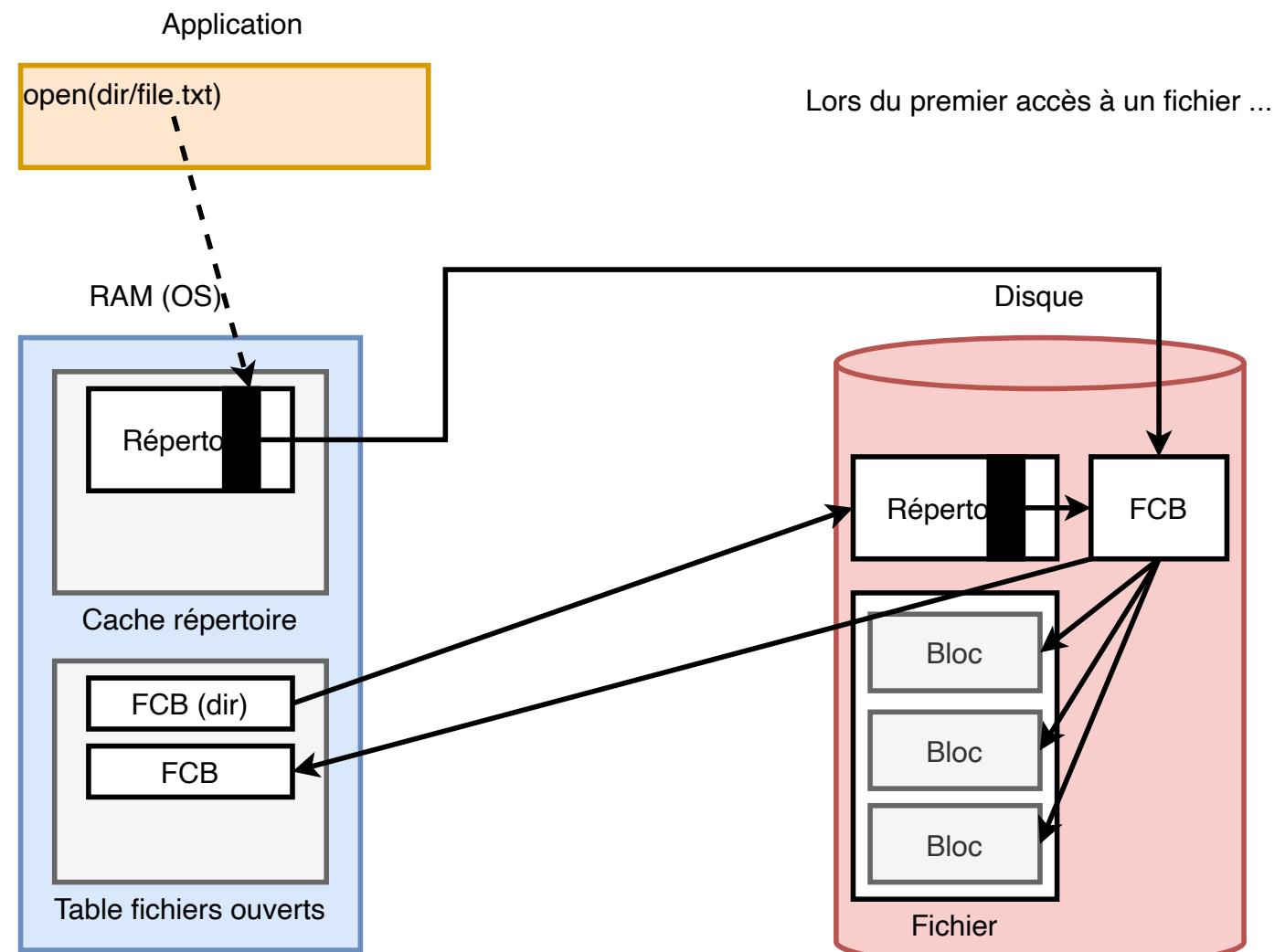
ACCÈS À UN FICHIER



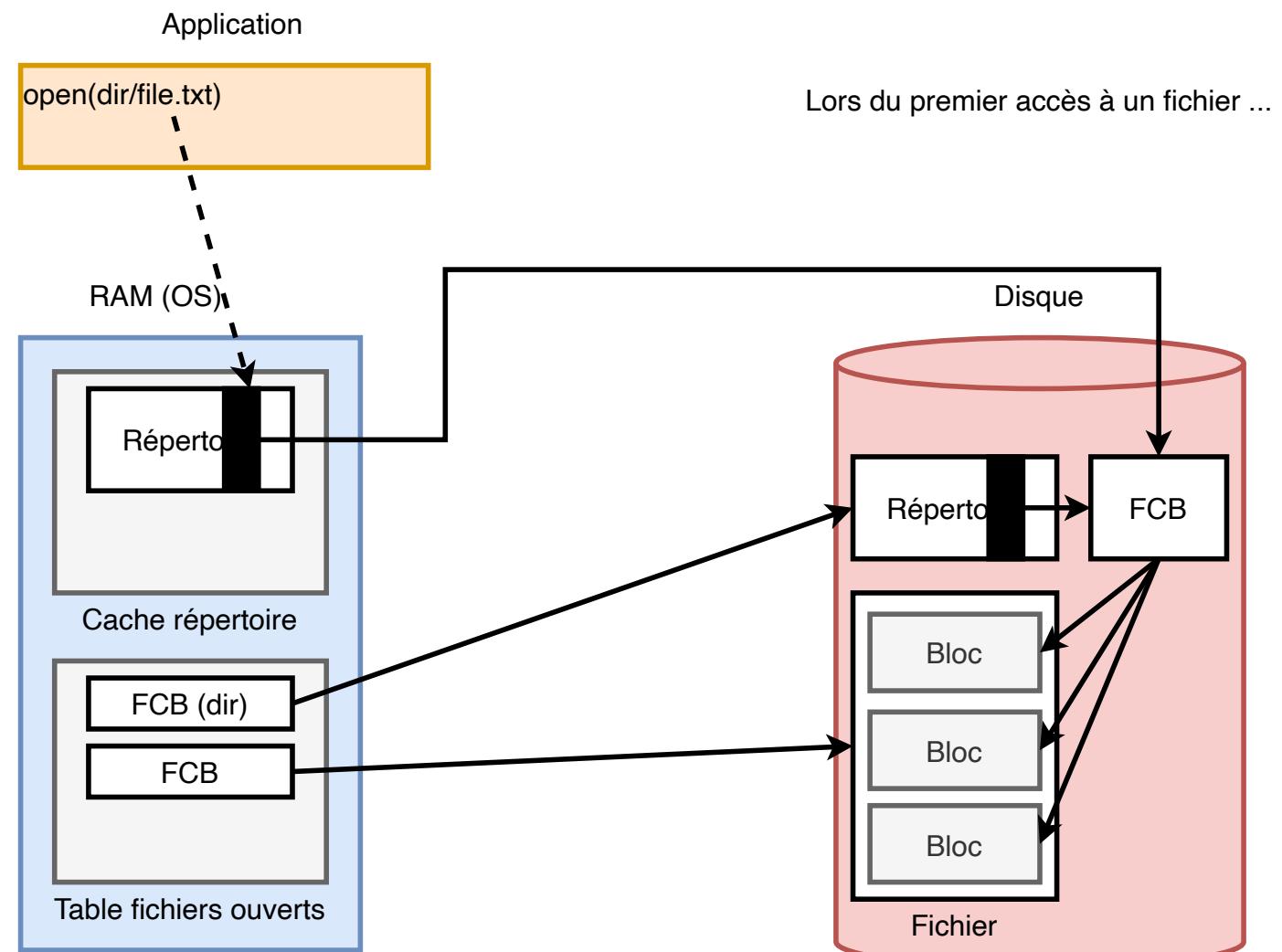
ACCÈS À UN FICHIER



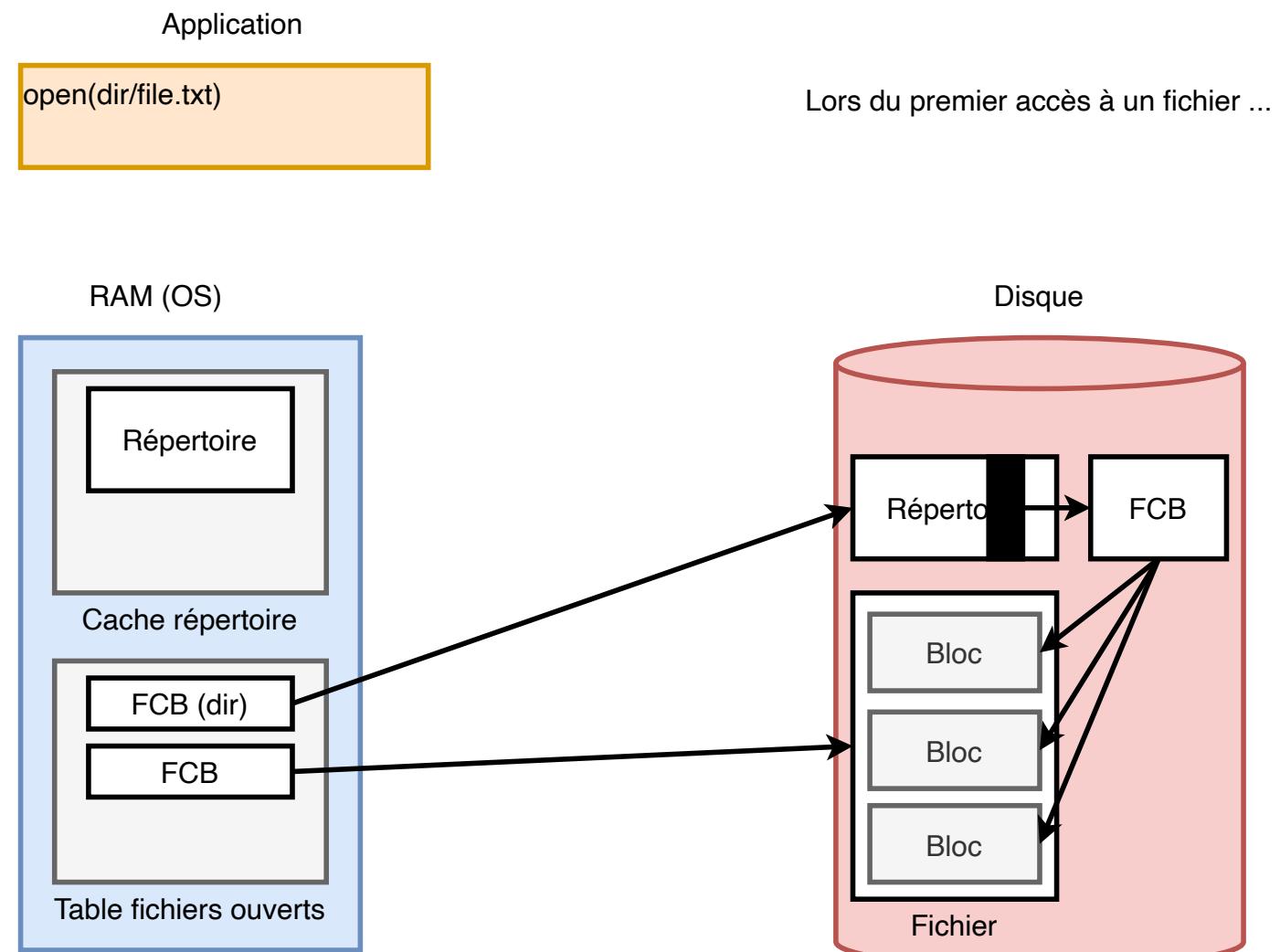
ACCÈS À UN FICHIER



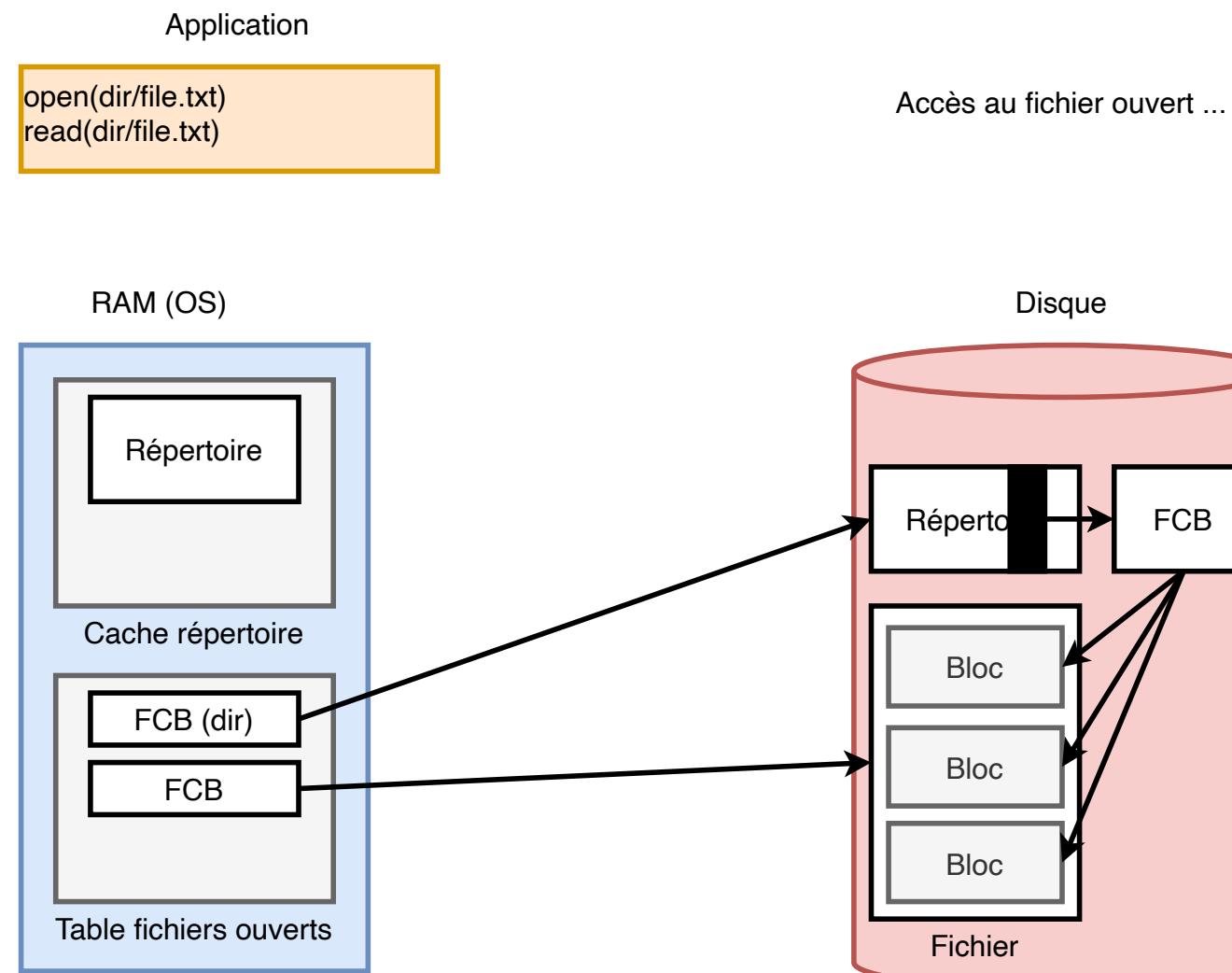
ACCÈS À UN FICHIER



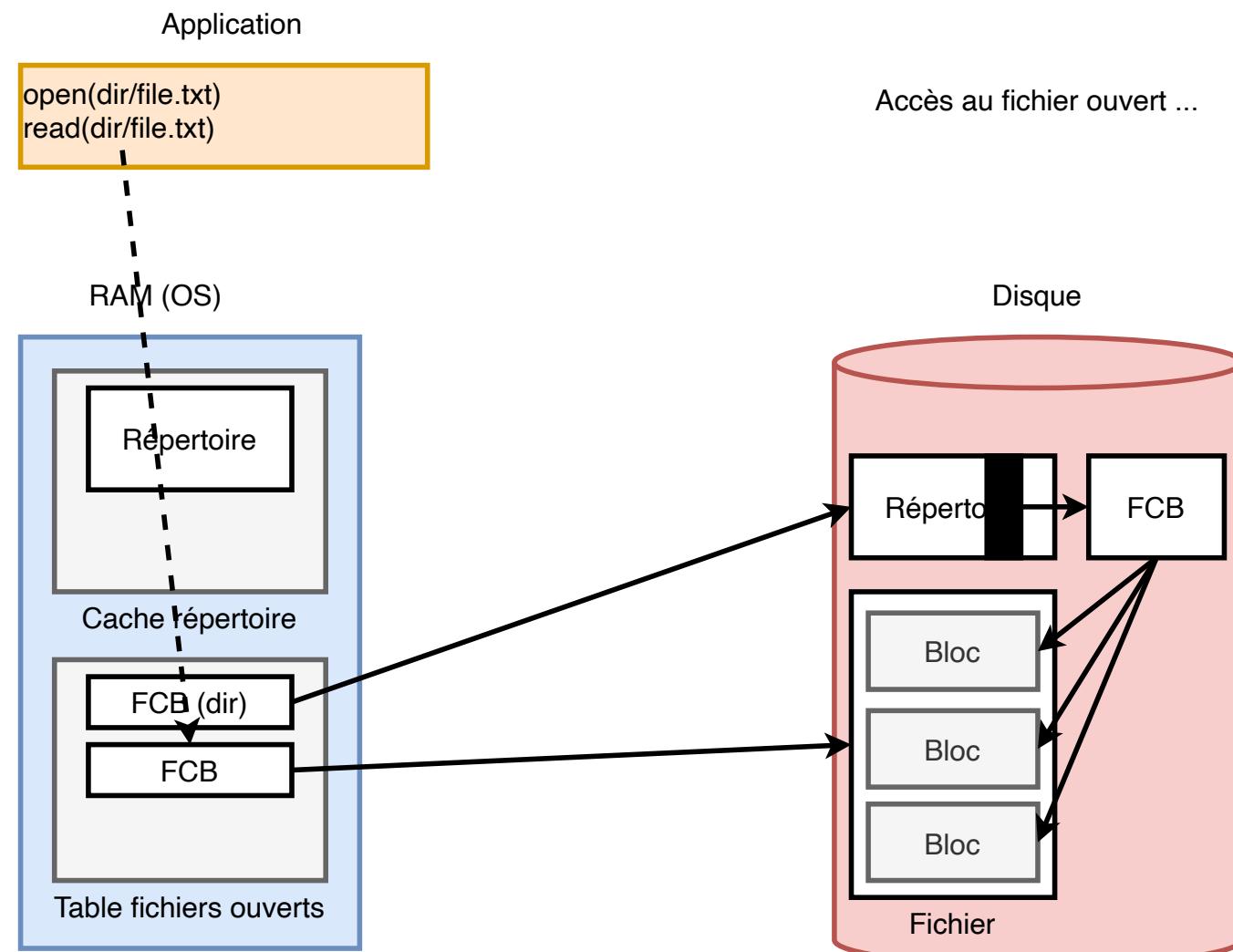
ACCÈS À UN FICHIER



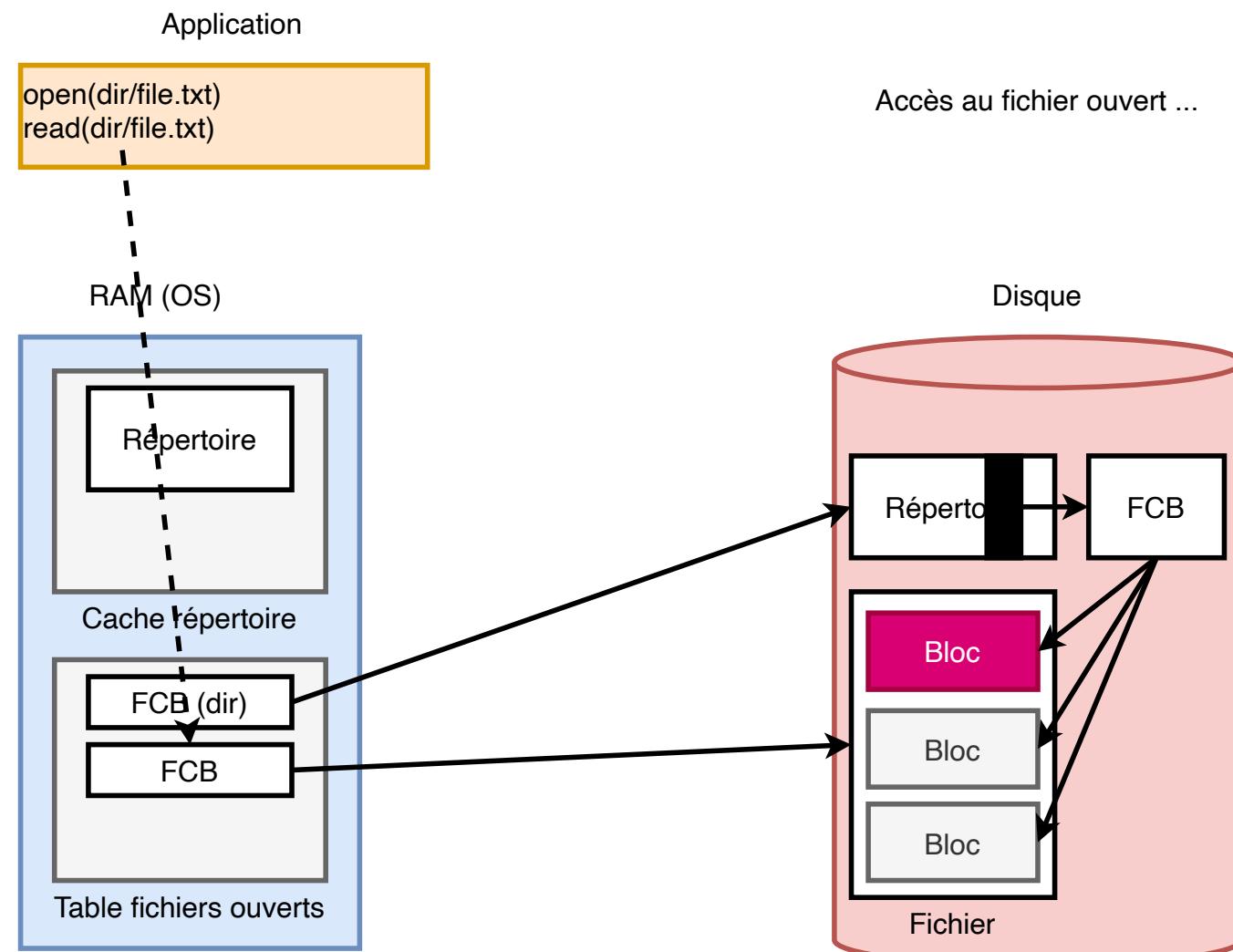
ACCÈS À UN FICHIER



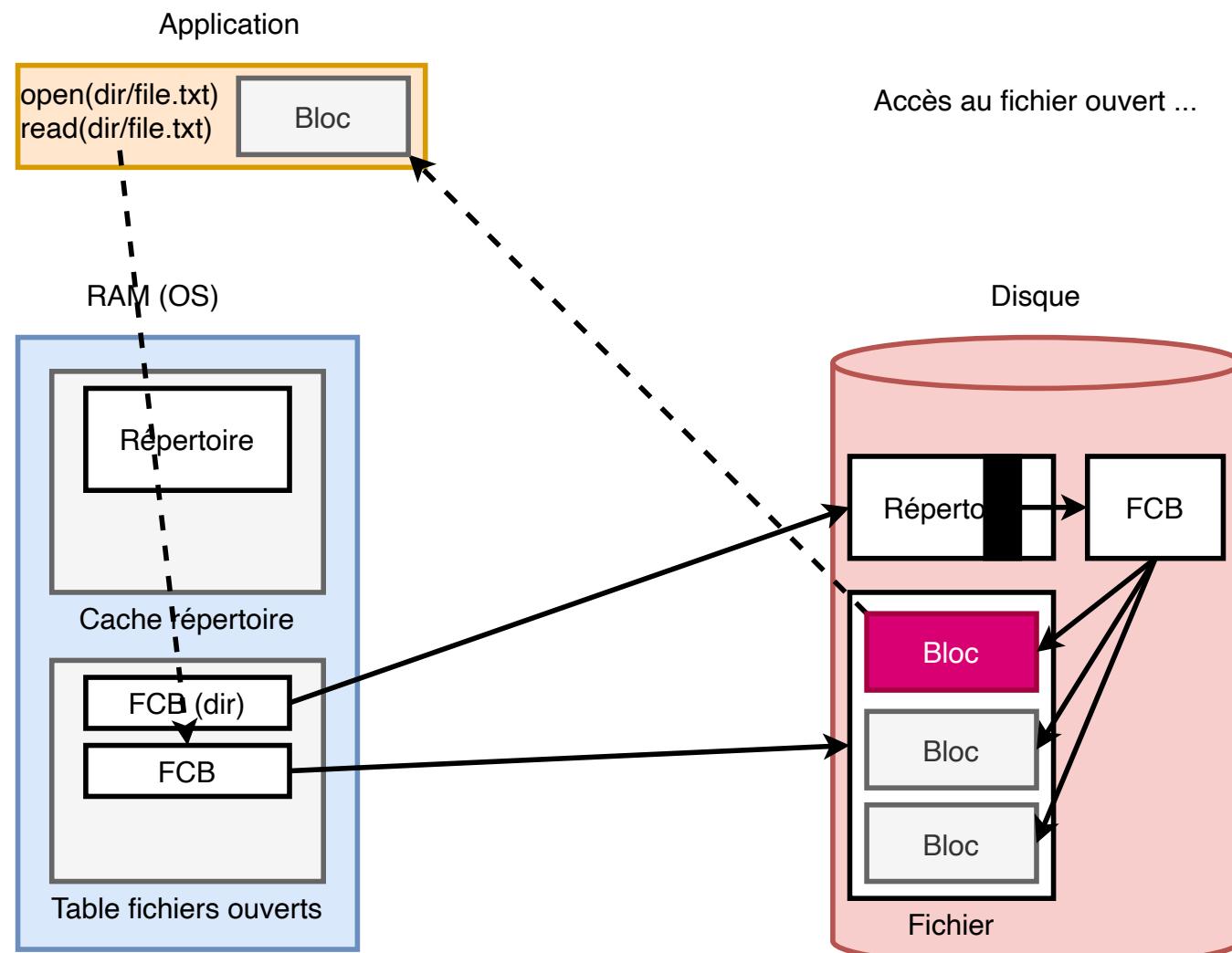
ACCÈS À UN FICHIER



ACCÈS À UN FICHIER



ACCÈS À UN FICHIER



STRUCTURES SUR DISQUE

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)
 - 👉 disque flash (lecture directe rapide)

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)
 - 👉 disque flash (lecture directe rapide)
 - **Critères d'optimisation**

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)
 - 👉 disque flash (lecture directe rapide)
 - **Critères d'optimisation**
 - 👉 temps d'accès séquentiel

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)
 - 👉 disque flash (lecture directe rapide)
 - **Critères d'optimisation**
 - 👉 temps d'accès séquentiel
 - 👉 temps d'accès direct

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)
 - 👉 disque flash (lecture directe rapide)
 - **Critères d'optimisation**
 - 👉 temps d'accès séquentiel
 - 👉 temps d'accès direct
 - 👉 espace

STRUCTURES SUR DISQUE

- Suivant le **support physique** et le **critère que l'on souhaite optimiser**, différentes stratégies de stockage sont possibles.
 - **Supports physiques**
 - 👉 CD-R (pas de modification)
 - 👉 bande (lecture séquentielle)
 - 👉 disque dur (lecture séquentielle à privilégier)
 - 👉 disque flash (lecture directe rapide)
 - **Critères d'optimisation**
 - 👉 temps d'accès séquentiel
 - 👉 temps d'accès direct
 - 👉 espace
 - 👉 évolution des fichiers (changement de taille, suppression, ajout)

ALLOCATION

ALLOCATION

Problème

ALLOCATION

Problème

- Fichiers → blocs logiques → blocs physiques

ALLOCATION

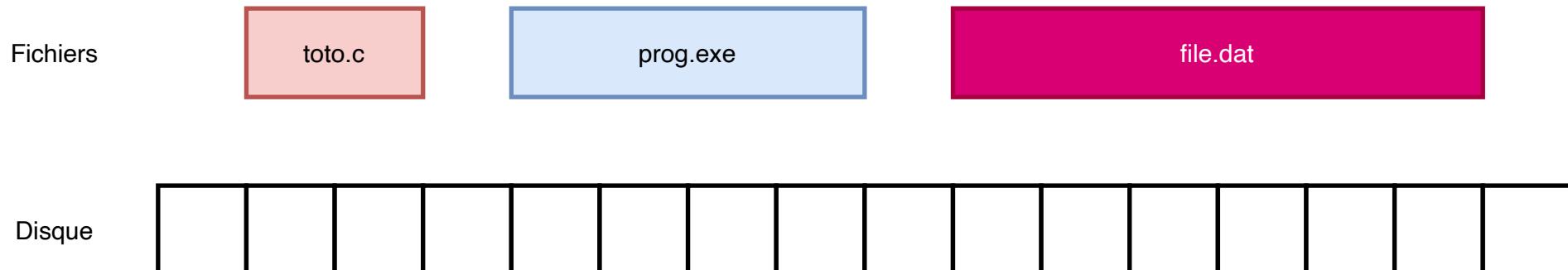
Problème

- Fichiers → blocs logiques → blocs physiques
👉 choix des blocs physiques pour 1 fichier donné

ALLOCATION

Problème

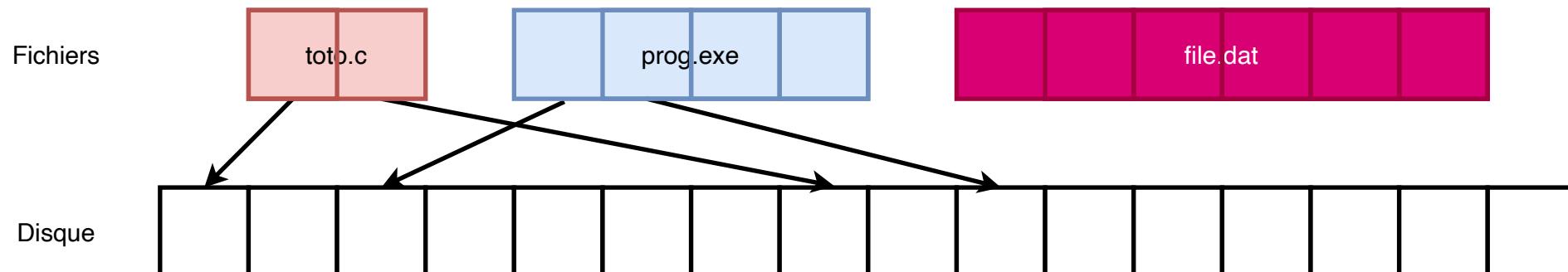
- Fichiers → blocs logiques → blocs physiques
👉 choix des blocs physiques pour 1 fichier donné



ALLOCATION

Problème

- Fichiers → blocs logiques → blocs physiques
👉 choix des blocs physiques pour 1 fichier donné



ALLOCATION

- Trois méthodes d'allocation possibles:

ALLOCATION

- Trois méthodes d'allocation possibles:
 1. allocation contiguë

ALLOCATION

- Trois méthodes d'allocation possibles:
 1. allocation contiguë
 2. allocation chaînée

ALLOCATION

- **Trois méthodes d'allocation possibles:**
 1. allocation contiguë
 2. allocation chaînée
 3. allocation indexée

ALLOCATION

- Trois méthodes d'allocation possibles:
 1. allocation contiguë
 2. allocation chaînée
 3. allocation indexée
- Chaque méthode a ses **avantages** et ses **inconvénients** !

FILE ALLOCATION TABLE (FAT)

FILE ALLOCATION TABLE (FAT)

Principe

FILE ALLOCATION TABLE (FAT)

Principe

- Allocation indexée

FILE ALLOCATION TABLE (FAT)

Principe

- Allocation **indexée**
- Liste chaînée des **index** des blocs en **début de partition**

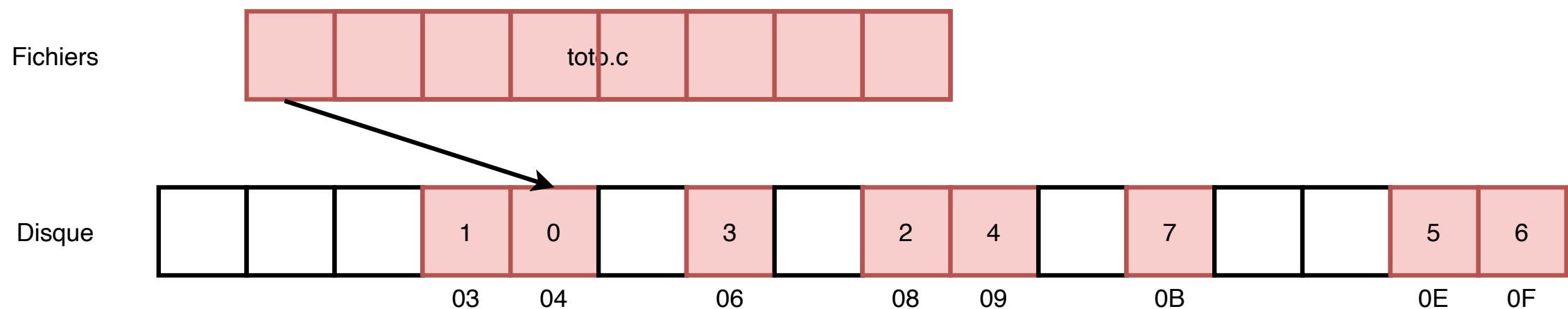
FILE ALLOCATION TABLE (FAT)

Principe

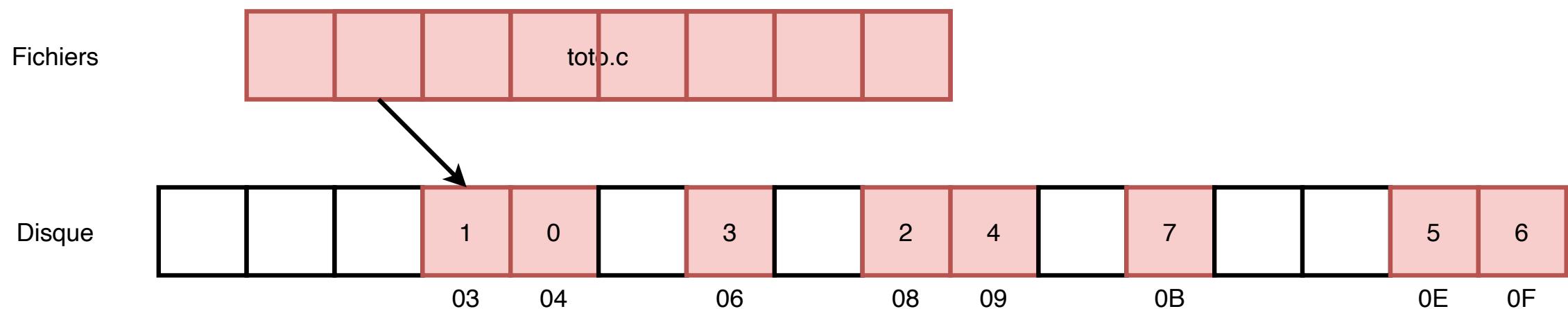
- Allocation **indexée**
- Liste chaînée des **index** des blocs en **début de partition**
- Utilisé sous **MSDOS (Intel)** et **OS/2 (IBM)**

FILE ALLOCATION TABLE (FAT)

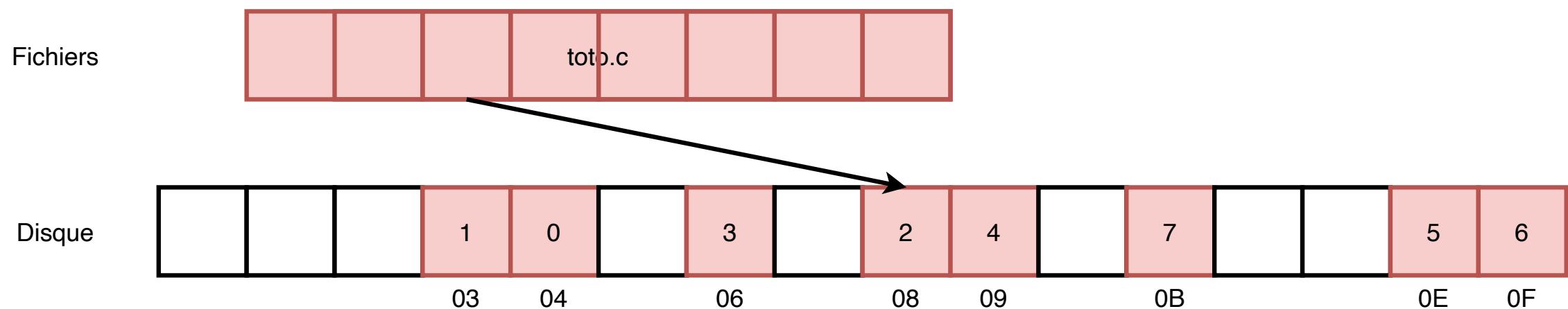
FILE ALLOCATION TABLE (FAT)



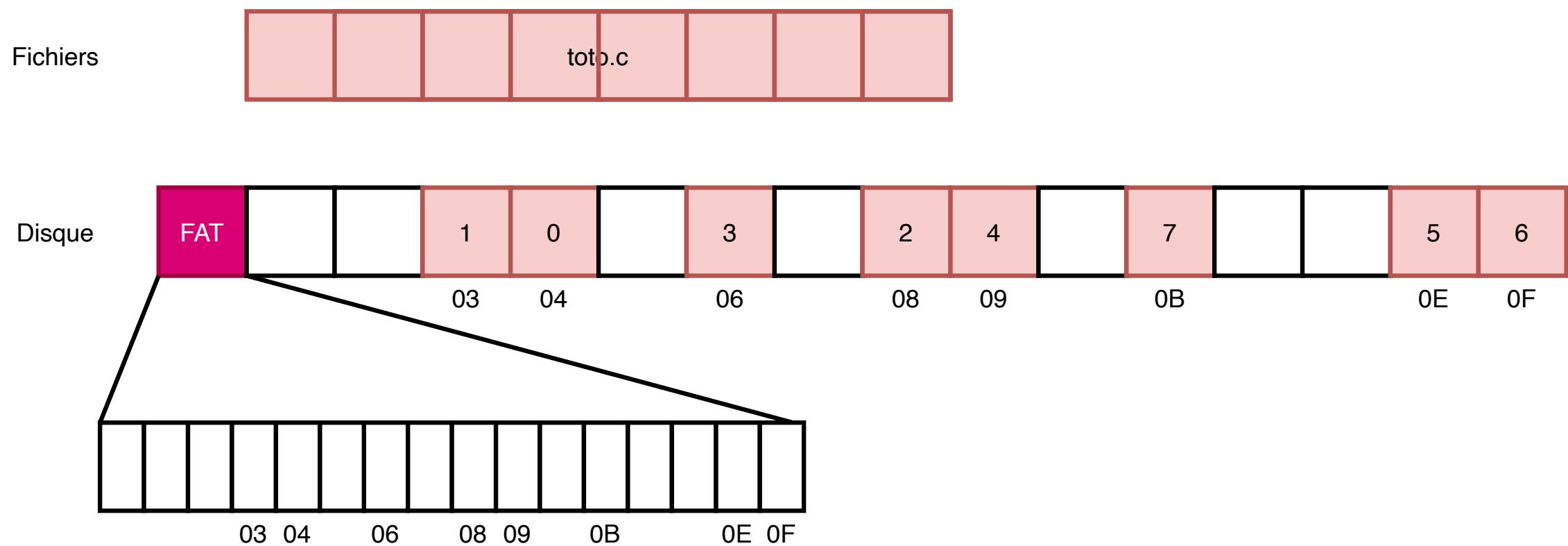
FILE ALLOCATION TABLE (FAT)



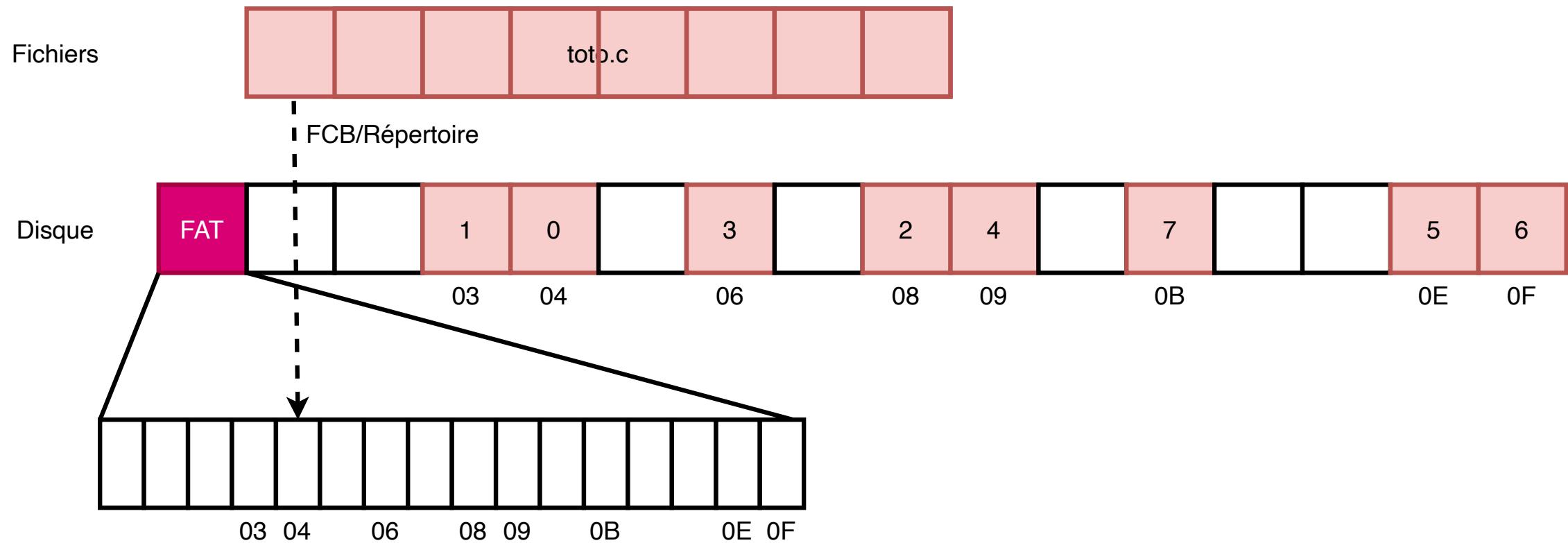
FILE ALLOCATION TABLE (FAT)



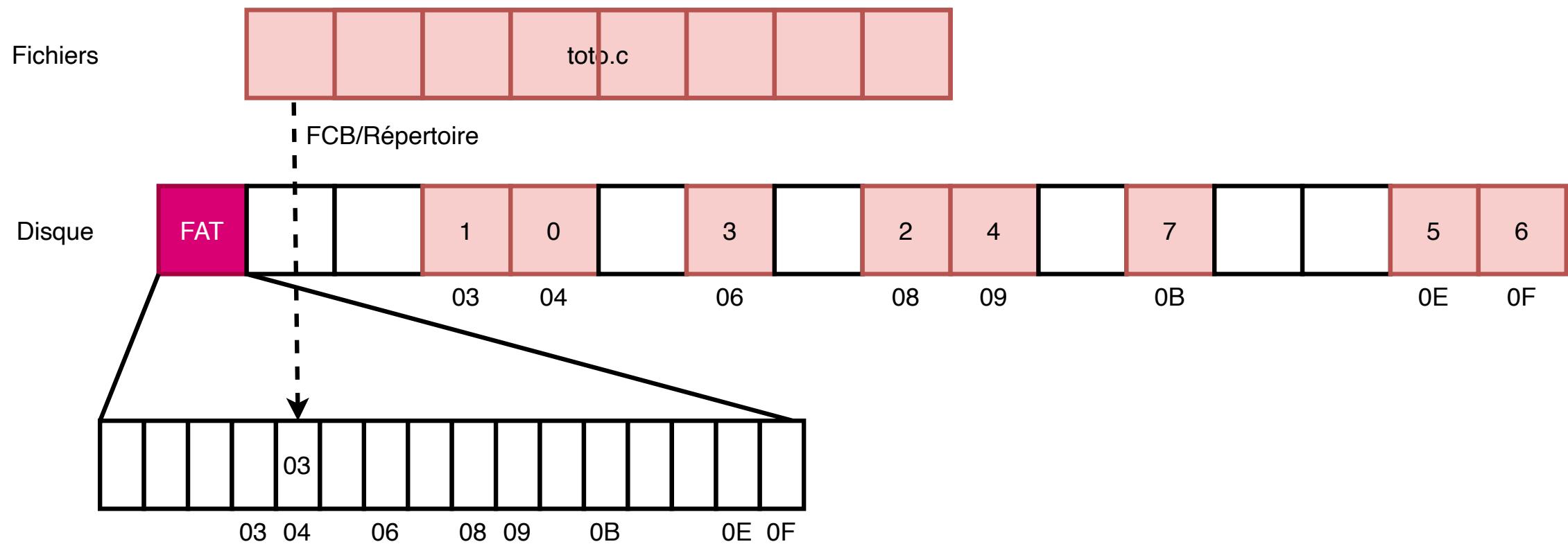
FILE ALLOCATION TABLE (FAT)



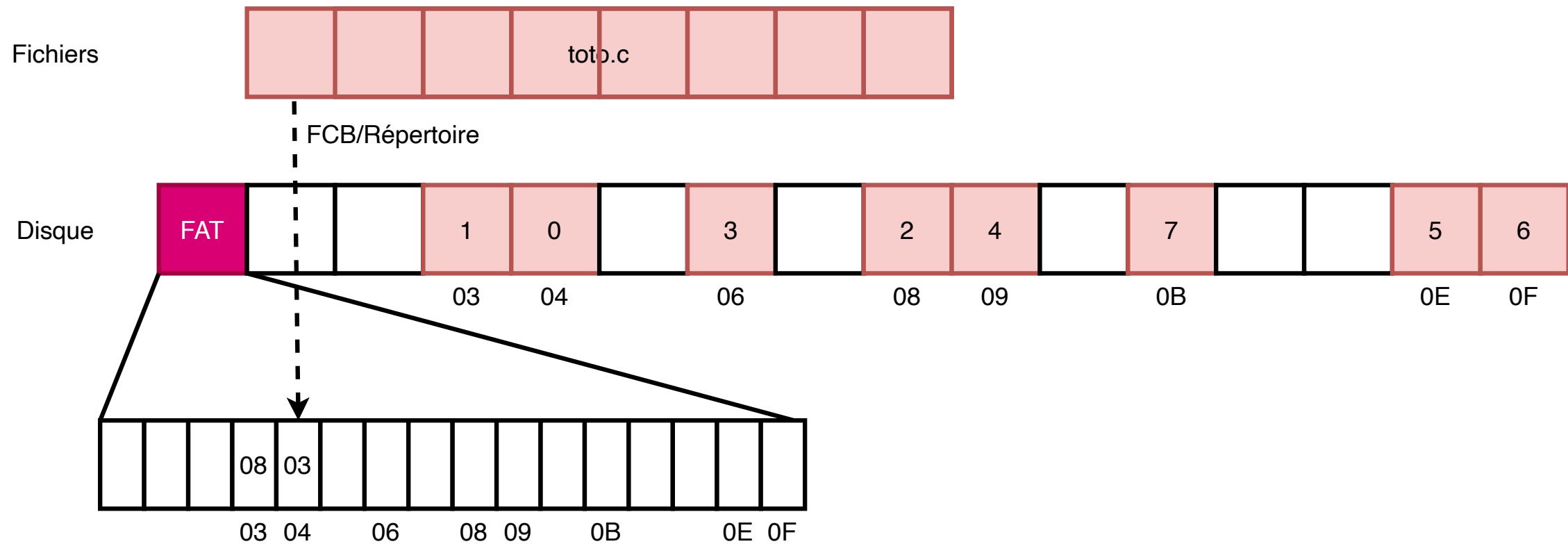
FILE ALLOCATION TABLE (FAT)



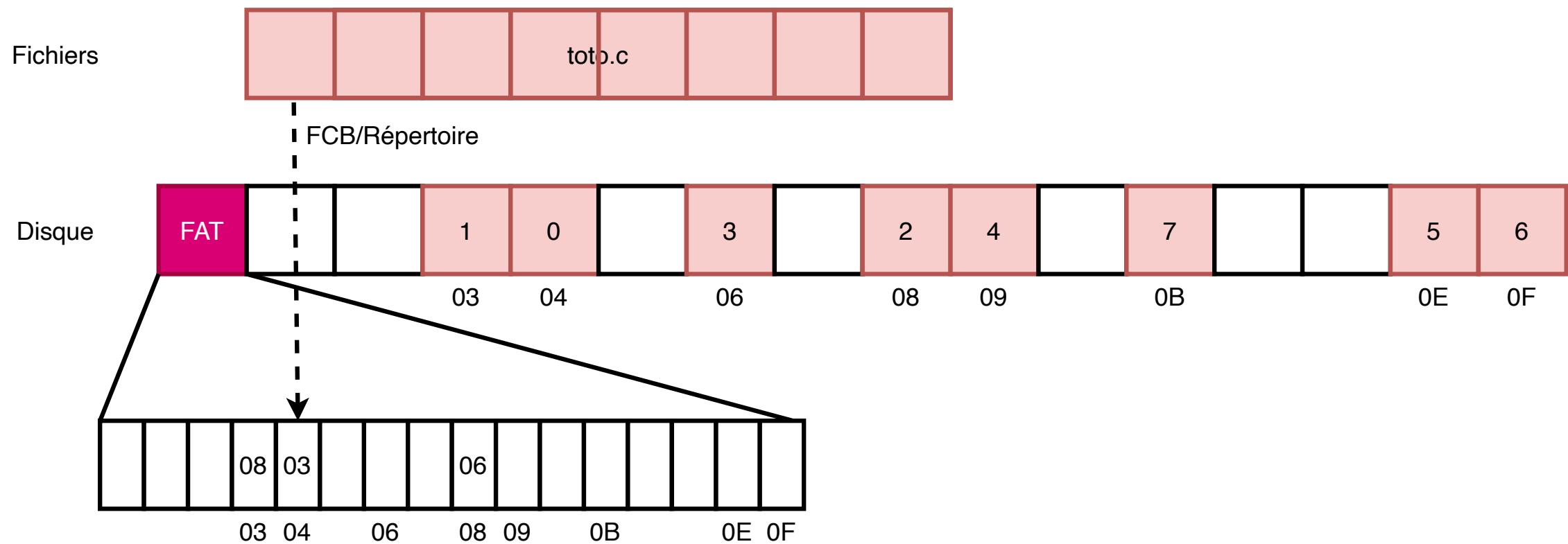
FILE ALLOCATION TABLE (FAT)



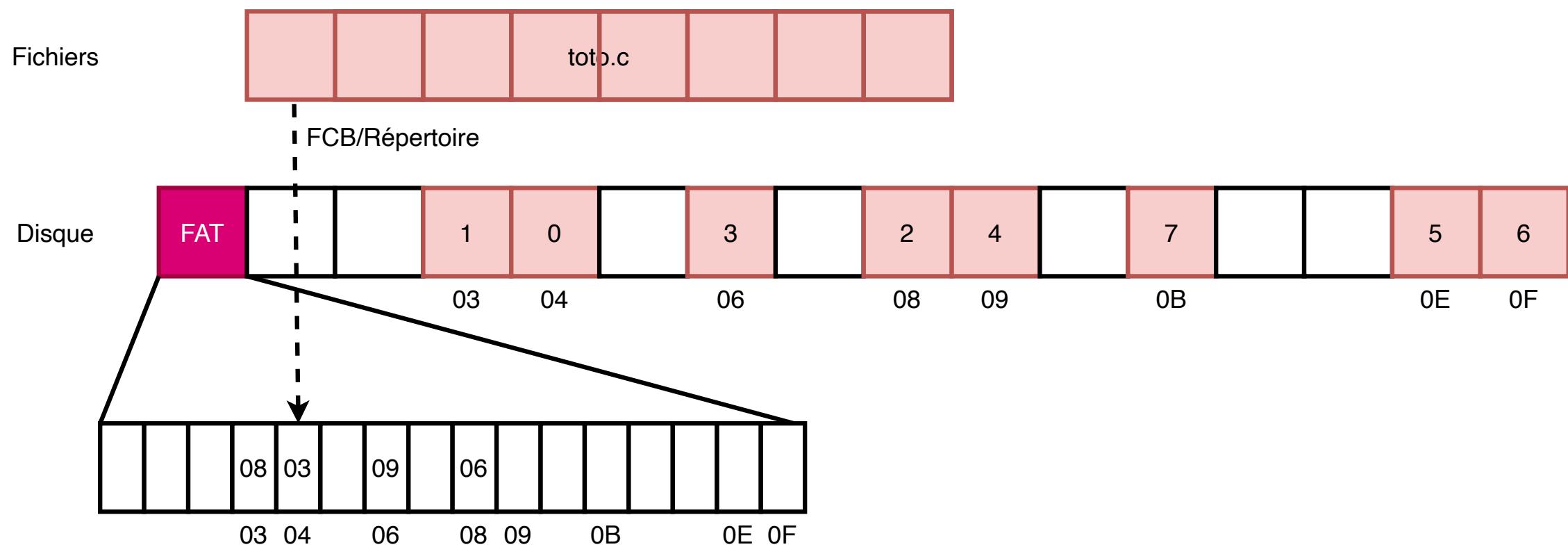
FILE ALLOCATION TABLE (FAT)



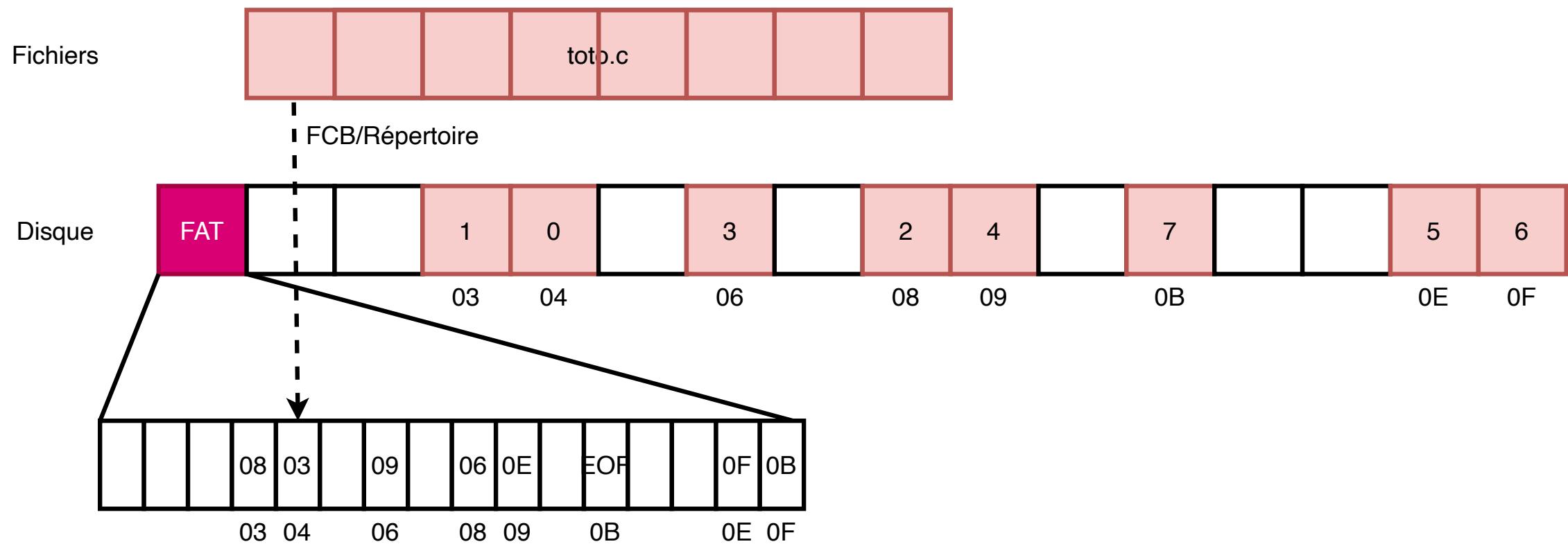
FILE ALLOCATION TABLE (FAT)



FILE ALLOCATION TABLE (FAT)



FILE ALLOCATION TABLE (FAT)



FILE ALLOCATION TABLE (FAT)

FILE ALLOCATION TABLE (FAT)

- Avantages

FILE ALLOCATION TABLE (FAT)

- **Avantages**
 - ✓ FCB : adresse premier bloc = premier index

FILE ALLOCATION TABLE (FAT)

- **Avantages**
 - ✓ FCB : adresse premier bloc = premier index
 - ✓ Pas de fragmentation ([allocation indexée](#))

FILE ALLOCATION TABLE (FAT)

- **Avantages**
 - ✓ FCB : adresse premier bloc = premier index
 - ✓ Pas de fragmentation ([allocation indexée](#))
 - ✓ Allocation bloc simple

FILE ALLOCATION TABLE (FAT)

- **Avantages**

- ✓ **FCB** : adresse premier bloc = premier index
- ✓ Pas de fragmentation ([allocation indexée](#))
- ✓ Allocation bloc simple
- ✓ Accès rapide ([FAT](#) chargée en cache puis accès direct disque)

FILE ALLOCATION TABLE (FAT)

- **Avantages**
 - ✓ FCB : adresse premier bloc = premier index
 - ✓ Pas de fragmentation ([allocation indexée](#))
 - ✓ Allocation bloc simple
 - ✓ Accès rapide ([FAT](#) chargée en cache puis accès direct disque)
- **Inconvénients**

FILE ALLOCATION TABLE (FAT)

- **Avantages**
 - ✓ FCB : adresse premier bloc = premier index
 - ✓ Pas de fragmentation ([allocation indexée](#))
 - ✓ Allocation bloc simple
 - ✓ Accès rapide ([FAT](#) chargée en cache puis accès direct disque)
- **Inconvénients**
 - ✗ Fiabilité : [FAT](#) perdue → disque foutu! ☹

FILE ALLOCATION TABLE (FAT)

- **Avantages**

- ✓ **FCB** : adresse premier bloc = premier index
- ✓ Pas de fragmentation (**allocation indexée**)
- ✓ Allocation bloc simple
- ✓ Accès rapide (**FAT** chargée en cache puis accès direct disque)

- **Inconvénients**

- ✗ Fiabilité : **FAT** perdue → disque foutu! ☹
 - 👉 doubler la **FAT** (sur 2 blocs distincts)

SCHÉMA COMBINÉ

SCHÉMA COMBINÉ

Principe

SCHÉMA COMBINÉ

Principe

- Combine l'**allocation chaînée** et l'**allocation indexée**

SCHÉMA COMBINÉ

Principe

- Combine l'**allocation chaînée** et l'**allocation indexée**
- Index = k premiers blocs du fichier + $(n - k)$ blocs d'indirection

SCHÉMA COMBINÉ

Principe

- Combine l'**allocation chaînée** et l'**allocation indexée**
- Index = k premiers blocs du fichier + $(n - k)$ blocs d'indirection
 - ✓ moins de perte pour les petits fichiers

SCHÉMA COMBINÉ

Principe

- Combine l'**allocation chaînée** et l'**allocation indexée**
- Index = k premiers blocs du fichier + $(n - k)$ blocs d'indirection
 - ✓ moins de perte pour les petits fichiers
 - ✓ accès rapide

SCHÉMA COMBINÉ

Principe

- Combine l'**allocation chaînée** et l'**allocation indexée**
- Index = k premiers blocs du fichier + $(n - k)$ blocs d'indirection
 - ✓ moins de perte pour les petits fichiers
 - ✓ accès rapide
- Utilisé par **Linux (EXTFS)**

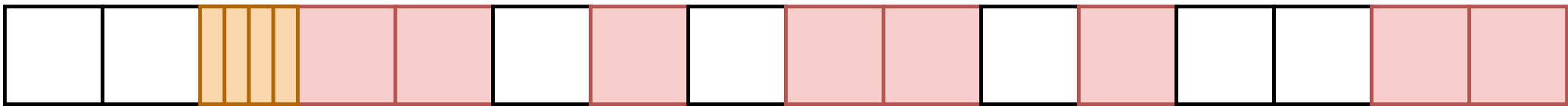
SCHÉMA COMBINÉ

SCHÉMA COMBINÉ

Fichiers



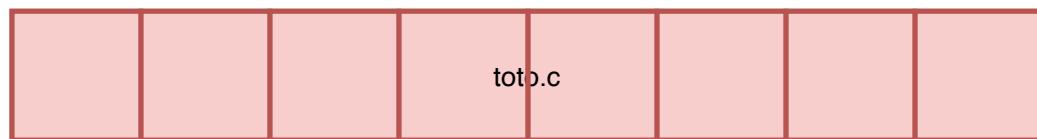
Disque



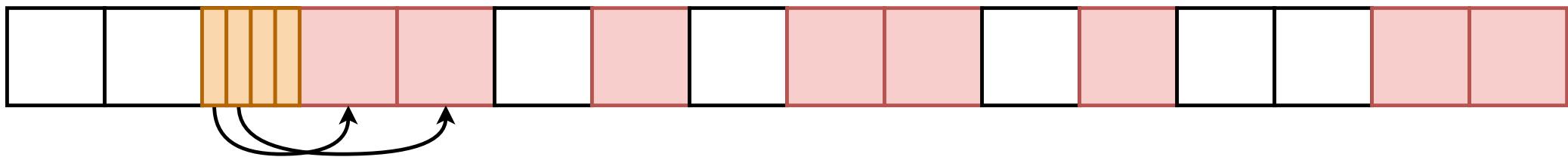
Ici, $k=2$ (et $n=4$)

SCHÉMA COMBINÉ

Fichiers



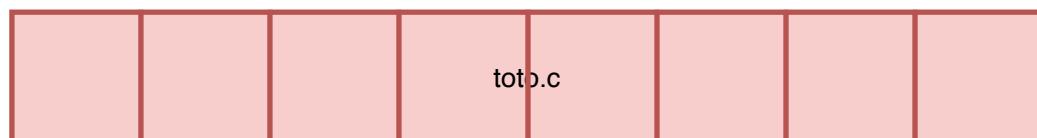
Disque



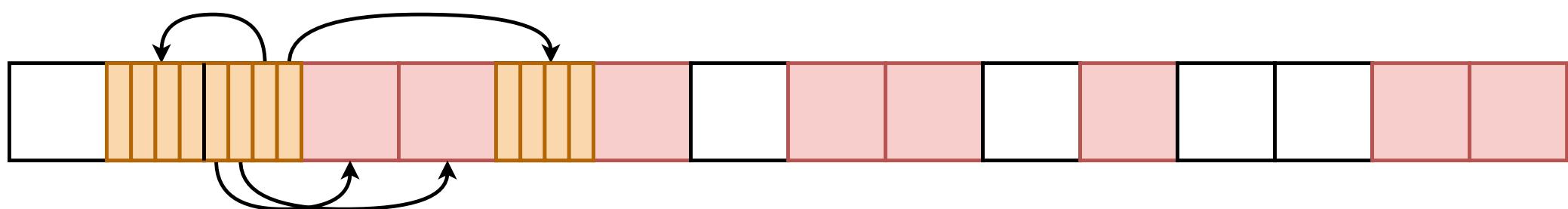
Ici, $k=2$ (et $n=4$)

SCHÉMA COMBINÉ

Fichiers



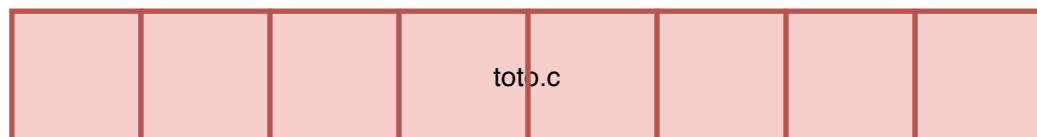
Disque



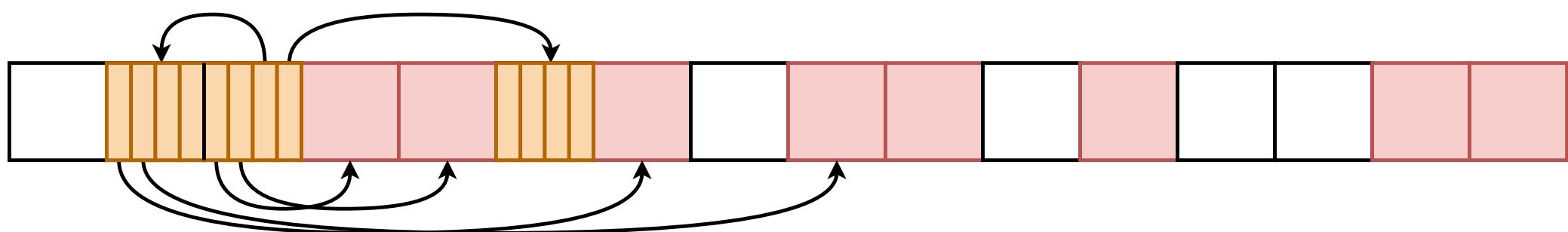
Ici, $k=2$ (et $n=4$)

SCHÉMA COMBINÉ

Fichiers



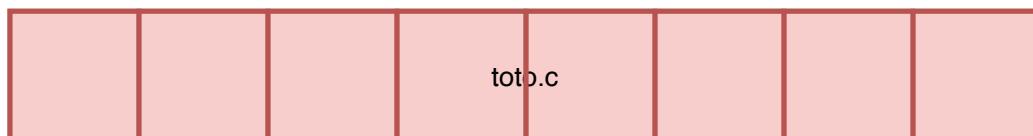
Disque



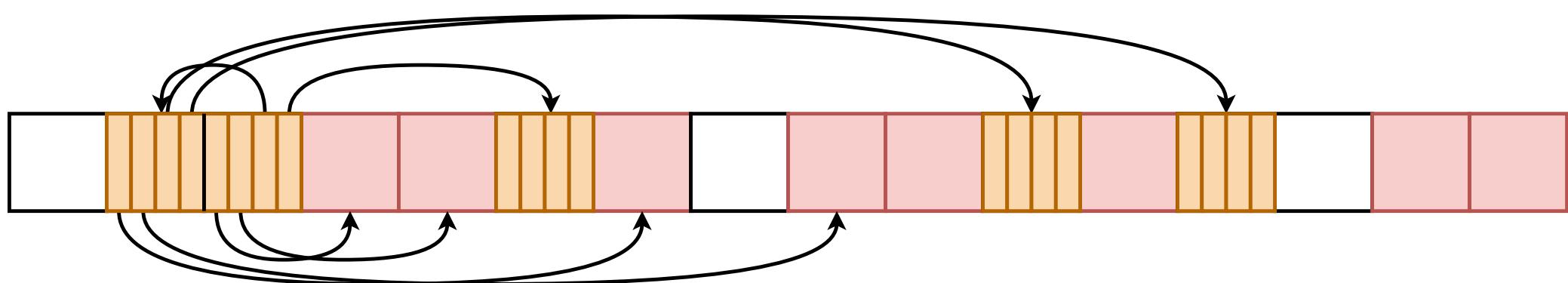
Ici, $k=2$ (et $n=4$)

SCHÉMA COMBINÉ

Fichiers



Disque



Ici, $k=2$ (et $n=4$)

FREE SPACE MANAGEMENT

FREE SPACE MANAGEMENT

- **Bit vector / bit map** : contient 1 bit par bloc (positionné à 1 si vide)

FREE SPACE MANAGEMENT

- **Bit vector / bit map** : contient 1 bit par bloc (positionné à 1 si vide)
 - Peu volumineux en **RAM**

FREE SPACE MANAGEMENT

- **Bit vector / bit map** : contient 1 bit par bloc (positionné à 1 si vide)
 - Peu volumineux en **RAM**
 - Rapide à chercher

FREE SPACE MANAGEMENT

- **Bit vector / bit map** : contient 1 bit par bloc (positionné à 1 si vide)
 - Peu volumineux en **RAM**
 - Rapide à chercher
 - Ex: **ext2, ext3** (Recherche de bloc libres par groupes de 8 si possible et proches des autres blocs du fichier)

FREE SPACE MANAGEMENT

- **Bit vector / bit map** : contient 1 bit par bloc (positionné à 1 si vide)
 - Peu volumineux en **RAM**
 - Rapide à chercher
 - Ex: **ext2, ext3** (Recherche de bloc libres par groupes de 8 si possible et proches des autres blocs du fichier)
- **Liste chaînée**

FREE SPACE MANAGEMENT

- **Bit vector / bit map** : contient 1 bit par bloc (positionné à 1 si vide)
 - Peu volumineux en **RAM**
 - Rapide à chercher
 - Ex: **ext2, ext3** (Recherche de bloc libres par groupes de 8 si possible et proches des autres blocs du fichier)
- **Liste chaînée**
 - La **FAT** possède **un bit free** pour chaque bloc et rend donc le service de gestion d'espace libre

AUTRES SERVICES DES FS

AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Journalisation**
 - Actions d'écritures par transaction (**commit/rollback** en cas de problème)
 - Exemple : **NTFS, ext4**

AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Journalisation**
 - Actions d'écritures par transaction (**commit/rollback** en cas de problème)
 - Exemple : **NTFS, ext4**
- **Chiffrement**
 - Exemple : Windows Encrypted FileSystem (**EFS**)
 - Exemple : Linux **ext4, DM-Crypt LUKS** (bas niveau)

AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Journalisation**
 - Actions d'écritures par transaction (**commit/rollback** en cas de problème)
 - Exemple : **NTFS, ext4**
- **Chiffrement**
 - Exemple : Windows Encrypted FileSystem (**EFS**)
 - Exemple : Linux **ext4, DM-Crypt LUKS** (bas niveau)
- **Compression**
 - Exemple : **ZFS, BrtFS, NTFS**

AUTRES SERVICES DES FS

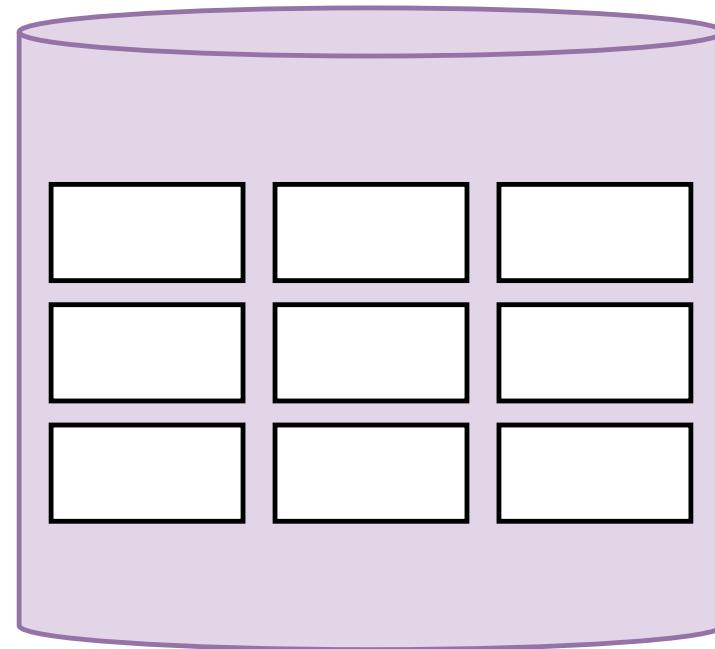
Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Déduplication**
 - Exemple : [BrtFS](#), [SDFS](#), [ZFS](#)

AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

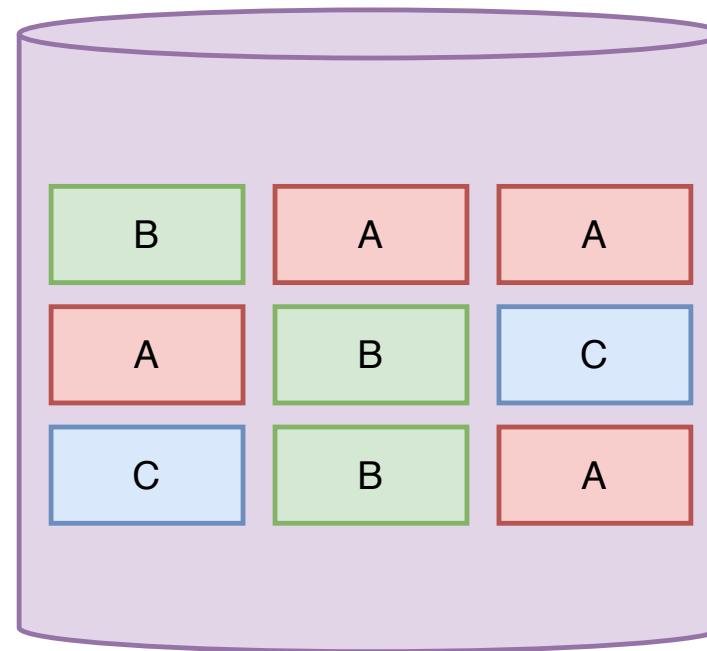
- **Déduplication**
 - Exemple : [BrtFS](#), [SDFS](#), [ZFS](#)



AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

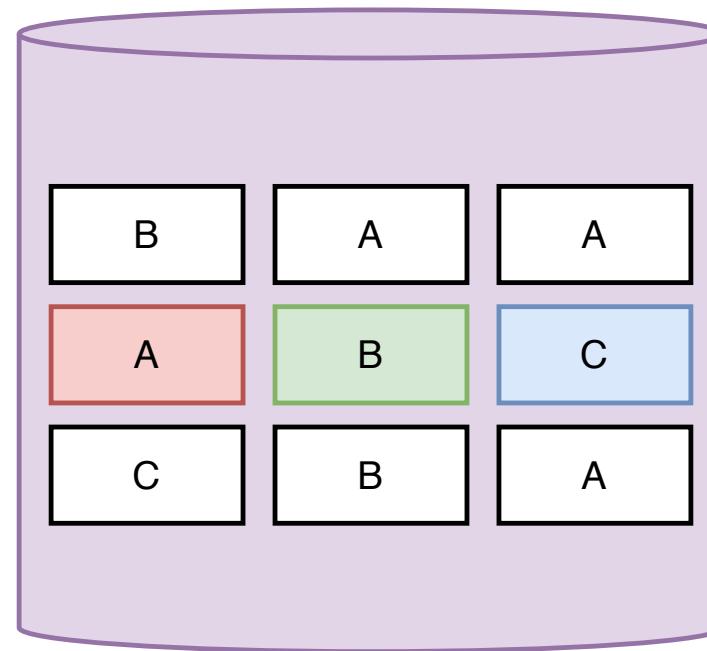
- **Déduplication**
 - Exemple : [BrtFS](#), [SDFS](#), [ZFS](#)



AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Déduplication**
 - Exemple : [BrtFS](#), [SDFS](#), [ZFS](#)



AUTRES SERVICES DES FS

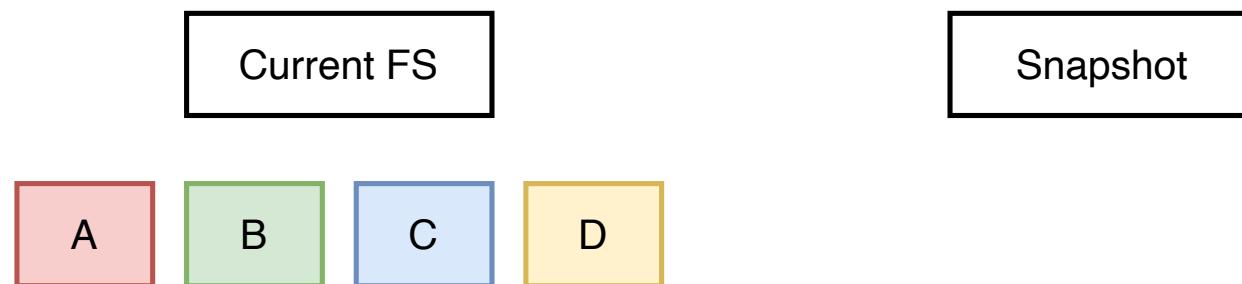
Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Clichés instantanés (snapshots)**
 - Technologie de **Copy-On-Write**
 - Exemple : **NTFS** (Volume Shadow Copy), Linux **LVM**
 - **Extension** : gestion de versions (snapshots continus)
 - Exemple : **ext3cow**, **tux3**, **NILFS**

AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

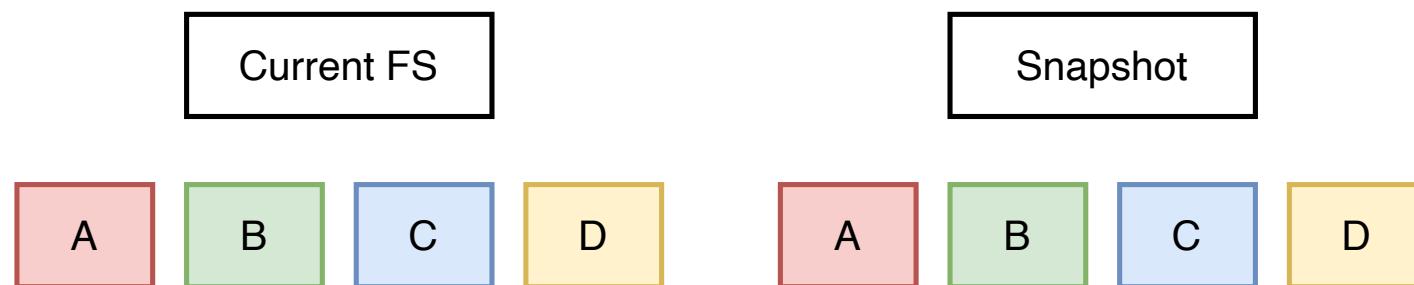
- **Clichés instantanés (snapshots)**
 - Technologie de **Copy-On-Write**
 - Exemple : **NTFS** (Volume Shadow Copy), Linux **LVM**
 - **Extension** : gestion de versions (snapshots continus)
 - Exemple : **ext3cow**, **tux3**, **NILFS**



AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

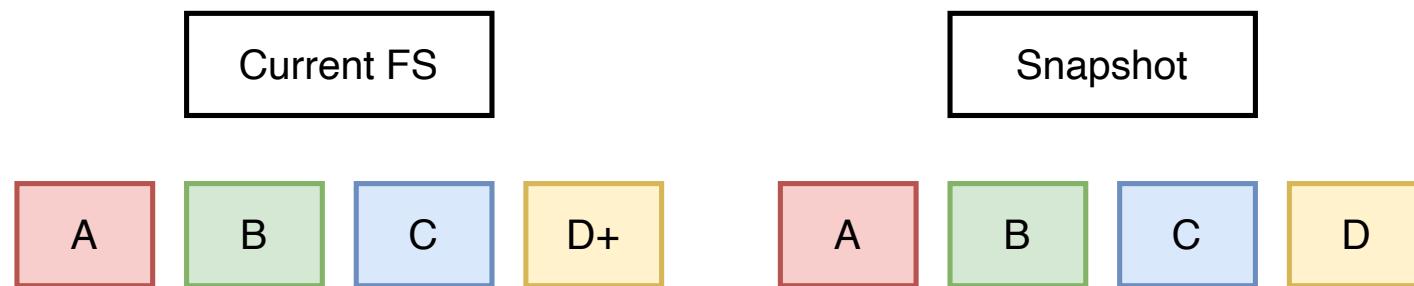
- **Clichés instantanés (snapshots)**
 - Technologie de **Copy-On-Write**
 - Exemple : **NTFS** (Volume Shadow Copy), Linux **LVM**
 - **Extension** : gestion de versions (snapshots continus)
 - Exemple : **ext3cow**, **tux3**, **NILFS**



AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

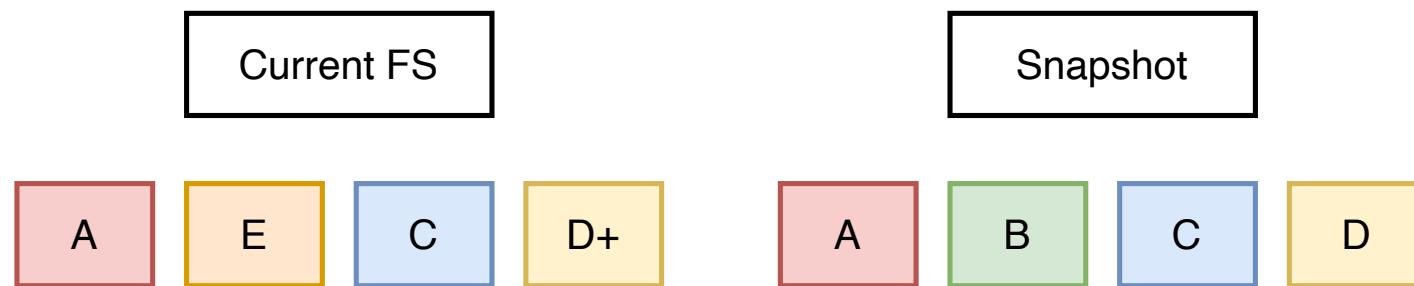
- **Clichés instantanés (snapshots)**
 - Technologie de **Copy-On-Write**
 - Exemple : **NTFS** (Volume Shadow Copy), Linux **LVM**
 - **Extension** : gestion de versions (snapshots continus)
 - Exemple : **ext3cow**, **tux3**, **NILFS**



AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

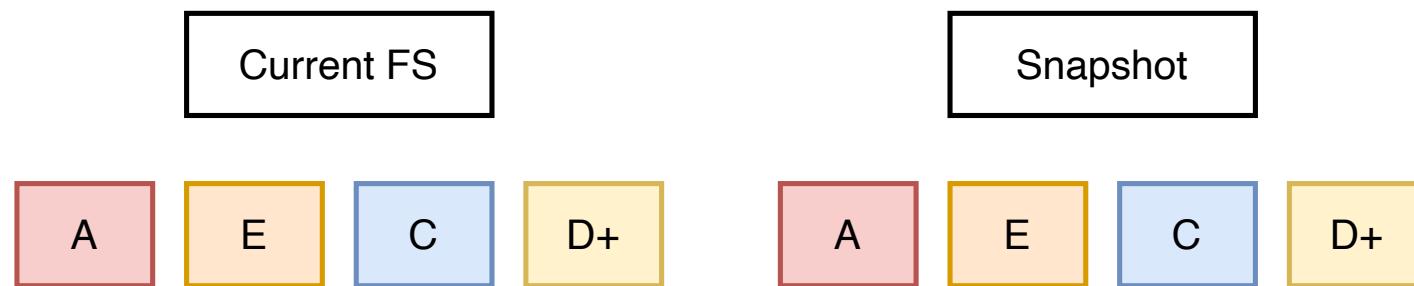
- **Clichés instantanés (snapshots)**
 - Technologie de **Copy-On-Write**
 - Exemple : **NTFS** (Volume Shadow Copy), Linux **LVM**
 - **Extension** : gestion de versions (snapshots continus)
 - Exemple : **ext3cow**, **tux3**, **NILFS**



AUTRES SERVICES DES FS

Les systèmes de fichiers récents peuvent offrir des services supplémentaires:

- **Clichés instantanés (snapshots)**
 - Technologie de **Copy-On-Write**
 - Exemple : **NTFS** (Volume Shadow Copy), Linux **LVM**
 - **Extension** : gestion de versions (snapshots continus)
 - Exemple : **ext3cow**, **tux3**, **NILFS**



VOLUME LOGIQUE

La gestion par **volume logique (LVM)**, est une méthode d'allocation d'espace disque plus flexible que le partitionnement classique.

VOLUME LOGIQUE

La gestion par **volume logique (LVM)**, est une méthode d'allocation d'espace disque plus flexible que le partitionnement classique.

- Les **volumes physiques** sont regroupés en groupe de volumes équivalent à des pseudo disques durs.

VOLUME LOGIQUE

La gestion par **volume logique (LVM)**, est une méthode d'allocation d'espace disque plus flexible que le partitionnement classique.

- Les **volumes physiques** sont regroupés en groupe de volumes équivalent à des pseudo disques durs.
- Ils sont alors découpé en **volume logique** qui seront formatés et montés dans le **FS**.

VOLUME LOGIQUE

La gestion par **volume logique (LVM)**, est une méthode d'allocation d'espace disque plus flexible que le partitionnement classique.

- Les **volumes physiques** sont regroupés en groupe de volumes équivalent à des pseudo disques durs.
- Ils sont alors découpé en **volume logique** qui seront formatés et montés dans le **FS**.
- C'est l'une des manière de **virtualiser le stockage** qui permet beaucoup de flexibilité :

VOLUME LOGIQUE

La gestion par **volume logique (LVM)**, est une méthode d'allocation d'espace disque plus flexible que le partitionnement classique.

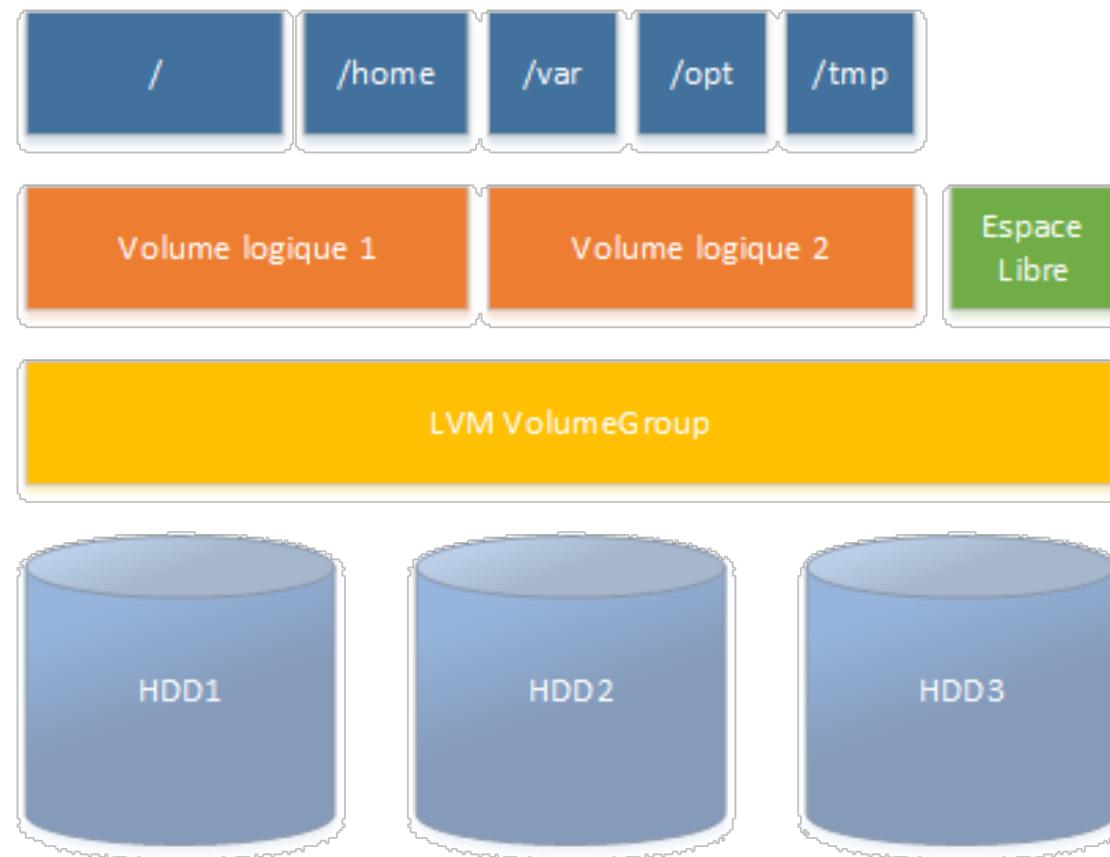
- Les **volumes physiques** sont regroupés en groupe de volumes équivalent à des pseudo disques durs.
- Ils sont alors découpé en **volume logique** qui seront formatés et montés dans le **FS**.
- C'est l'une des manière de **virtualiser le stockage** qui permet beaucoup de flexibilité :
👉 redimensionnement

VOLUME LOGIQUE

La gestion par **volume logique (LVM)**, est une méthode d'allocation d'espace disque plus flexible que le partitionnement classique.

- Les **volumes physiques** sont regroupés en groupe de volumes équivalent à des pseudo disques durs.
- Ils sont alors découpé en **volume logique** qui seront formatés et montés dans le **FS**.
- C'est l'une des manière de **virtualiser le stockage** qui permet beaucoup de flexibilité :
 - 👉 redimensionnement
 - 👉 tolérance aux pannes

VOLUME LOGIQUE



VOLUME LOGIQUE (RAID)

VOLUME LOGIQUE (RAID)

- Le **RAID** est un ensemble de **techniques de virtualisation** permettant de *répartir des données sur plusieurs disques durs*

VOLUME LOGIQUE (RAID)

- Le **RAID** est un ensemble de **techniques de virtualisation** permettant de *répartir des données sur plusieurs disques durs*
- Le **RAID** permet d'améliorer :

VOLUME LOGIQUE (RAID)

- Le **RAID** est un ensemble de **techniques de virtualisation** permettant de *répartir des données sur plusieurs disques durs*
- Le **RAID** permet d'améliorer :
 - 👉 les performances,

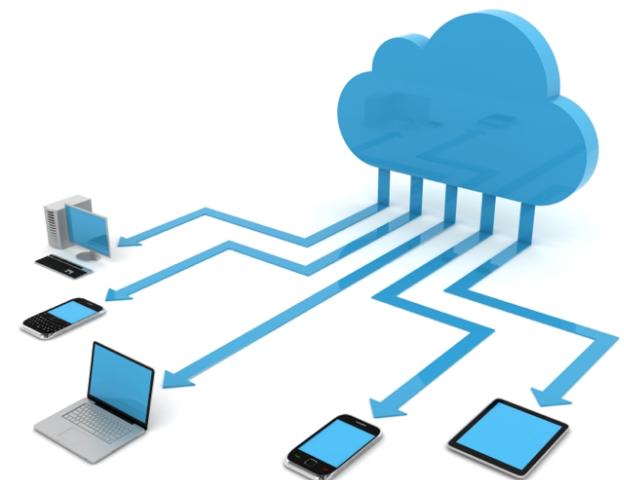
VOLUME LOGIQUE (RAID)

- Le **RAID** est un ensemble de **techniques de virtualisation** permettant de *répartir des données sur plusieurs disques durs*
- Le **RAID** permet d'améliorer :
 - 👉 les performances,
 - 👉 la sécurité

VOLUME LOGIQUE (RAID)

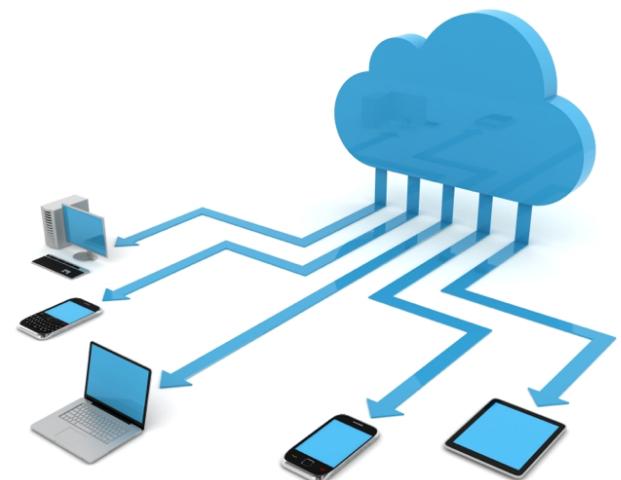
- Le **RAID** est un ensemble de **techniques de virtualisation** permettant de *répartir des données sur plusieurs disques durs*
- Le **RAID** permet d'améliorer :
 - 👉 les performances,
 - 👉 la sécurité
 - 👉 la tolérance aux pannes

LES SYSTÈMES DE FICHIER EN RÉSEAU



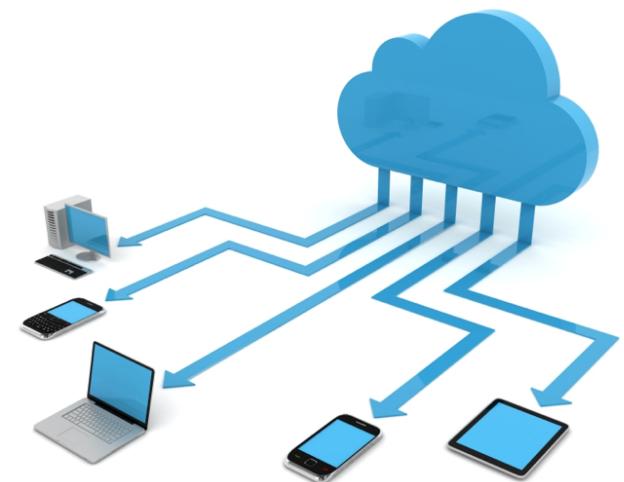
LES SYSTÈMES DE FICHIER EN RÉSEAU

- Présentent des **ressources de stockage réparties sur un réseau.**



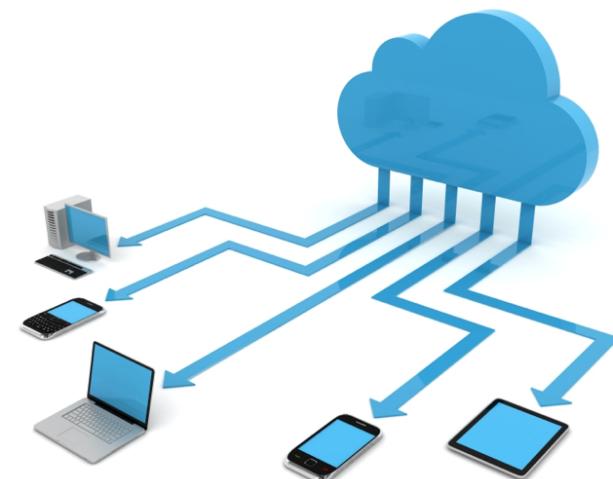
LES SYSTÈMES DE FICHIER EN RÉSEAU

- Présentent des **ressources de stockage réparties sur un réseau.**
- Masquent les **FS** sous-jacent
(Ex. : NFS de Unix, CIFS de Microsoft)



LES SYSTÈMES DE FICHIER EN RÉSEAU

- Présentent des **ressources de stockage réparties sur un réseau.**
- Masquent les **FS** sous-jacent
(Ex. : NFS de Unix, CIFS de Microsoft)
- Les nouvelles versions sont orientées **Cluster/Cloud**
(Ex. : Ceph, GlusterFS, GoogleFS/HadoopFS, RozoFS)



SÉCURITÉ

⌚ Fonctionnalités principales d'un OS

SÉCURITÉ INHÉRENTE À L'OS

SÉCURITÉ INHÉRENTE À L'OS

- Seul l'**OS** peut exécuter des **instructions privilégiées** (mode **Superviseur** du **CPU**)

SÉCURITÉ INHÉRENTE À L'OS

- Seul l'**OS** peut exécuter des **instructions privilégiées** (mode **Superviseur** du **CPU**)
 - Ex. : instructions de **bas niveau** (lecture/écritures de périphériques)

SÉCURITÉ INHÉRENTE À L'OS

- Seul l'**OS** peut exécuter des **instructions privilégiées** (mode **Superviseur du CPU**)
 - Ex. : instructions de **bas niveau** (lecture/écritures de périphériques)
- Les **espaces mémoires** de chaque processus et du noyau sont **isolés** (**pagination, segmentation**)

SÉCURITÉ INHÉRENTE À L'OS

- Seul l'**OS** peut exécuter des **instructions privilégiées** (mode **Superviseur** du **CPU**)
 - Ex. : instructions de **bas niveau** (lecture/écritures de périphériques)
- Les **espaces mémoires** de chaque processus et du noyau sont **isolés** (**pagination, segmentation**)
- Un processus monopolisant le **CPU** ne bloque pas le système

SÉCURITÉ INHÉRENTE À L'OS

- Seul l'**OS** peut exécuter des **instructions privilégiées** (mode **Superviseur du CPU**)
 - Ex. : instructions de **bas niveau** (lecture/écritures de périphériques)
- Les **espaces mémoires** de chaque processus et du noyau sont **isolés** (**pagination, segmentation**)
- Un processus monopolisant le **CPU** ne bloque pas le système
 - **Timer matériel** déclenchant la préemption du **CPU**

AUTRES MÉCANISMES

AUTRES MÉCANISMES

- OS multiutilisateurs

AUTRES MÉCANISMES

- OS multiutilisateurs
 - plusieurs identités

AUTRES MÉCANISMES

- OS multiutilisateurs
 - plusieurs identités
 - les processus appartiennent à une identité

AUTRES MÉCANISMES

- OS multiutilisateurs
 - plusieurs identités
 - les processus appartiennent à une identité
- Privilèges

AUTRES MÉCANISMES

- OS multiutilisateurs
 - plusieurs identités
 - les processus appartiennent à une identité
- Privilèges
 - des comptes avec des privilèges différents sur le système

AUTRES MÉCANISMES

- OS multiutilisateurs
 - plusieurs identités
 - les processus appartiennent à une identité
- Privilèges
 - des comptes avec des privilèges différents sur le système
 - 👉 root, administrator, ...

AUTRES MÉCANISMES

- OS multiutilisateurs
 - plusieurs identités
 - les processus appartiennent à une identité
- Privilèges
 - des comptes avec des privilèges différents sur le système
 - 👉 root, administrator, ...
 - des processus privilégiés avec des niveaux de privilèges granulaires

AUTRES MÉCANISMES

- **OS multiutilisateurs**
 - plusieurs identités
 - les processus appartiennent à une identité
- **Privilèges**
 - des comptes avec des privilèges différents sur le système
 - 👉 root, administrator, ...
 - des processus privilégiés avec des niveaux de privilèges granulaires
 - 👉 capabilities sous Linux par exemple

AUTRES MÉCANISMES

- **OS multiutilisateurs**
 - plusieurs identités
 - les processus appartiennent à une identité
- **Privilèges**
 - des comptes avec des privilèges différents sur le système
 - 👉 root, administrator, ...
 - des processus privilégiés avec des niveaux de privilèges granulaires
 - 👉 capabilities sous Linux par exemple
- **Droits sur des objets (DAC)**

AUTRES MÉCANISMES

- **OS multiutilisateurs**
 - plusieurs identités
 - les processus appartiennent à une identité
- **Privilèges**
 - des comptes avec des privilèges différents sur le système
 - 👉 root, administrator, ...
 - des processus privilégiés avec des niveaux de privilèges granulaires
 - 👉 capabilities sous Linux par exemple
- **Droits sur des objets (DAC)**
 - possibilité de positionner des droits sur les fichiers

AUTRES MÉCANISMES

- Contrôle d'accès Obligatoire (MAC)

AUTRES MÉCANISMES

- **Contrôle d'accès Obligatoire (MAC)**
 - implémente des règles gérées par un administrateur

AUTRES MÉCANISMES

- **Contrôle d'accès Obligatoire (MAC)**
 - implémente des règles gérées par un administrateur
 - 👉 Exemple: Selinux, Apparmor

AUTRES MÉCANISMES

- **Contrôle d'accès Obligatoire (MAC)**
 - implémente des règles gérées par un administrateur
 - 👉 Exemple: Selinux, Apparmor
- **NameSpaces, Cgroups, ...**

AUTRES MÉCANISMES

- **Contrôle d'accès Obligatoire (MAC)**
 - implémente des règles gérées par un administrateur
 - 👉 Exemple: Selinux, Apparmor
- **NameSpaces, Cgroups, ...**
 - On va en reparler très prochainement...

PLAN

- Architecture des ordinateurs
- Structure et évolution des OS
- Fonctionnalités principales d'un OS
- Virtualisation

[Retour au plan](#) - [Retour à l'accueil](#)

OS ET VIRTUALISATION

OS ET VIRTUALISATION

Le **système d'exploitation** abstrait les ressources d'exécution en utilisant différentes stratégies :

OS ET VIRTUALISATION

Le **système d'exploitation** abstrait les ressources d'exécution en utilisant différentes stratégies :

1. **Multiplexer** une ressource (partager son utilisation).
👉 CPU en temps partagé.

OS ET VIRTUALISATION

Le **système d'exploitation** abstrait les ressources d'exécution en utilisant différentes stratégies :

1. **Multiplexer** une ressource (partager son utilisation).
👉 **CPU** en temps partagé.
2. **Emuler** une ressource (créer une ressource purement logicielle).
👉 **Mémoire virtuelle** swappée.

OS ET VIRTUALISATION

Le **système d'exploitation** abstrait les ressources d'exécution en utilisant différentes stratégies :

1. **Multiplexer** une ressource (partager son utilisation).
👉 CPU en temps partagé.
2. **Emuler** une ressource (créer une ressource purement logicielle).
👉 Mémoire virtuelle swappée.
3. **Agréger** plusieurs ressources pour n'en présenter qu'une.
👉 arborescence du **Virtual FileSystem** et **Volumes logiques**

AVANT LA VIRTUALISATION

AVANT LA VIRTUALISATION

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance
 - 👉 impact d'une opération d'un service sur un autre

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance
 - 👉 impact d'une opération d'un service sur un autre
 - **solution** → limiter les dépendances entre applications

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance
 - 👉 impact d'une opération d'un service sur un autre
 - **solution** → limiter les dépendances entre applications
2. **Un serveur par application**

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance
 - 👉 impact d'une opération d'un service sur un autre
 - **solution** → limiter les dépendances entre applications
2. **Un serveur par application**
 - ✓ Bonne isolation,

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance
 - 👉 impact d'une opération d'un service sur un autre
 - **solution** → limiter les dépendances entre applications
2. **Un serveur par application**
 - ✓ Bonne isolation,
 - ✗ mais quel gâchis (HDD, RAM, CPU) !

Deux approches :

AVANT LA VIRTUALISATION

1. **Un gros serveur** portant de nombreuses applications
 - ✗ problèmes de dépendances sur des librairies
 - ✗ problèmes de maintenance
 - 👉 impact d'une opération d'un service sur un autre
 - **solution** → limiter les dépendances entre applications

2. **Un serveur par application**

- ✓ Bonne isolation,
- ✗ mais quel gâchis (HDD, RAM, CPU) !

Deux approches : • **solution** → consolidation de serveur

NIVEAUX DE VIRTUALISATION

NIVEAUX DE VIRTUALISATION

1. Machines virtuelles (émulation)

NIVEAUX DE VIRTUALISATION

1. Machines virtuelles (émulation)
2. Conteneurs

NIVEAUX DE VIRTUALISATION

1. Machines virtuelles (émulation)
2. Conteneurs
3. Applications cloud native

LES MACHINES VIRTUELLES (ÉMULATION)

⌚ Niveaux de virtualisation

LES MACHINES VIRTUELLES

Une machine virtuelle ([Virtual Machine - VM](#))

LES MACHINES VIRTUELLES

Une machine virtuelle (Virtual Machine - VM)

- un environnement qui fonctionne comme **un ordinateur virtuel** avec ses propres ressources

LES MACHINES VIRTUELLES

Une machine virtuelle (Virtual Machine - VM)

- un environnement qui fonctionne comme **un ordinateur virtuel** avec ses propres ressources (processeur, mémoire, disque, ...)

LES MACHINES VIRTUELLES

Une machine virtuelle (Virtual Machine - VM)

- un environnement qui fonctionne comme **un ordinateur virtuel** avec ses propres ressources (processeur, mémoire, disque, ...)

Les composants d'une VM

LES MACHINES VIRTUELLES

Une machine virtuelle (Virtual Machine - VM)

- un environnement qui fonctionne comme un ordinateur virtuel avec ses propres ressources (processeur, mémoire, disque, ...)

Les composants d'une VM

- une ou plusieurs applications.

LES MACHINES VIRTUELLES

Une machine virtuelle (Virtual Machine - VM)

- un environnement qui fonctionne comme un ordinateur virtuel avec ses propres ressources (processeur, mémoire, disque, ...)

Les composants d'une VM

- une ou plusieurs applications.
- un système d'exploitation.

LES MACHINES VIRTUELLES

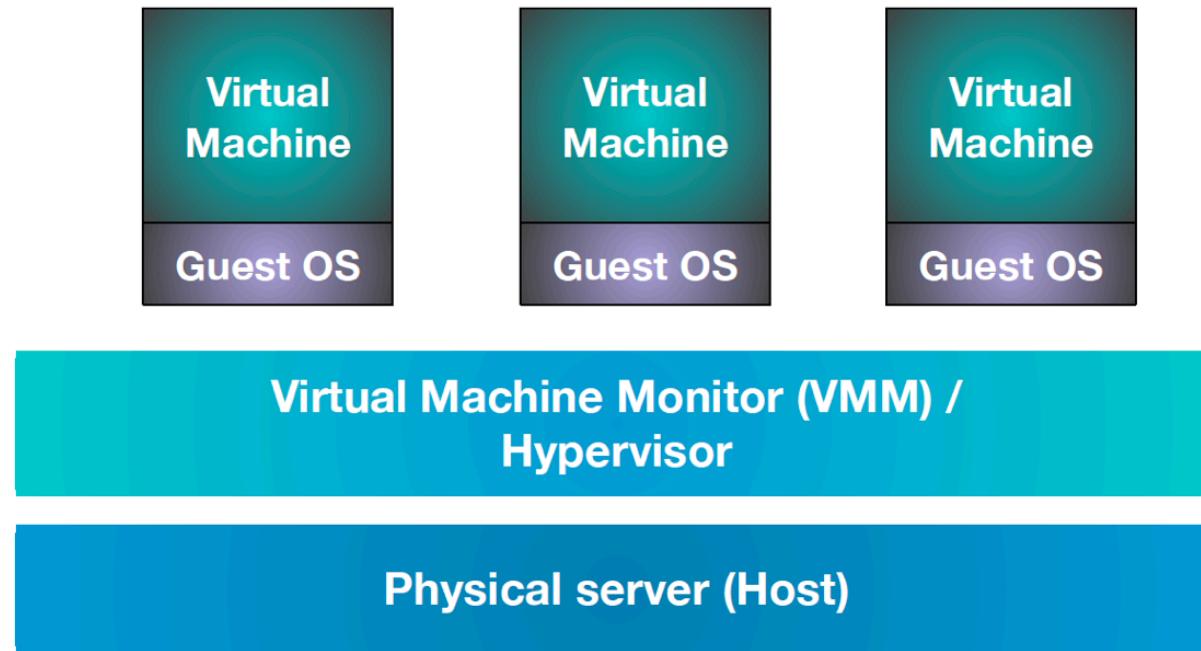
Une machine virtuelle (Virtual Machine - VM)

- un environnement qui fonctionne comme un ordinateur virtuel avec ses propres ressources (processeur, mémoire, disque, ...)

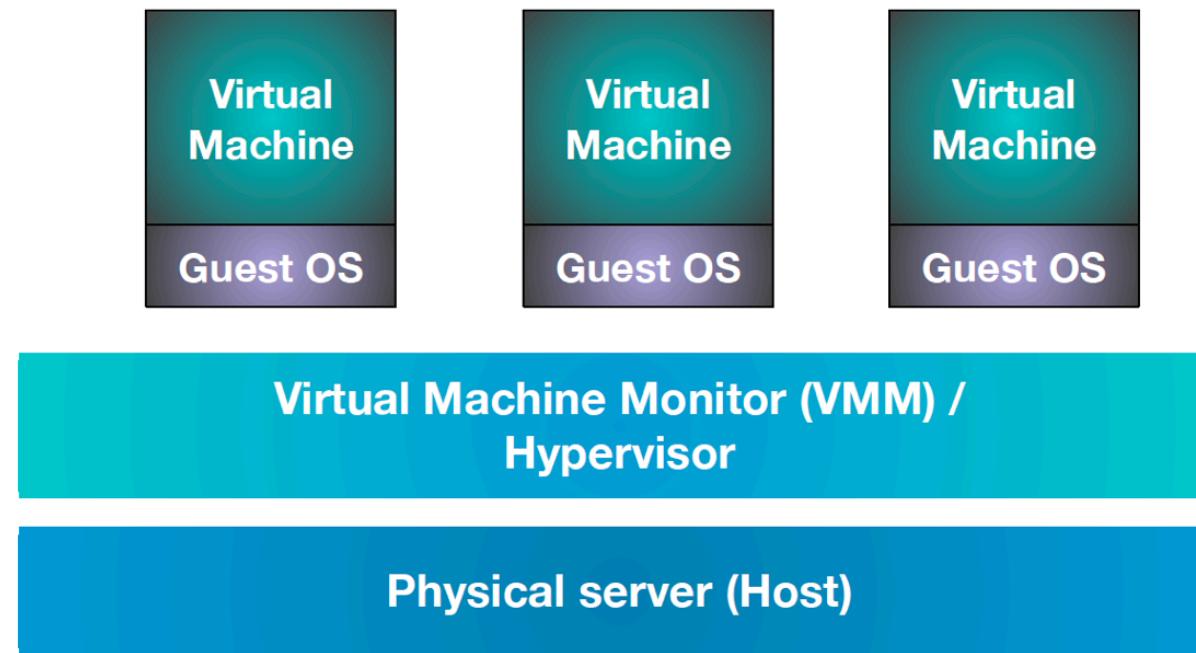
Les composants d'une VM

- une ou plusieurs applications.
- un système d'exploitation.
- un ensemble de périphériques virtuels.

LES HYPERVISEURS

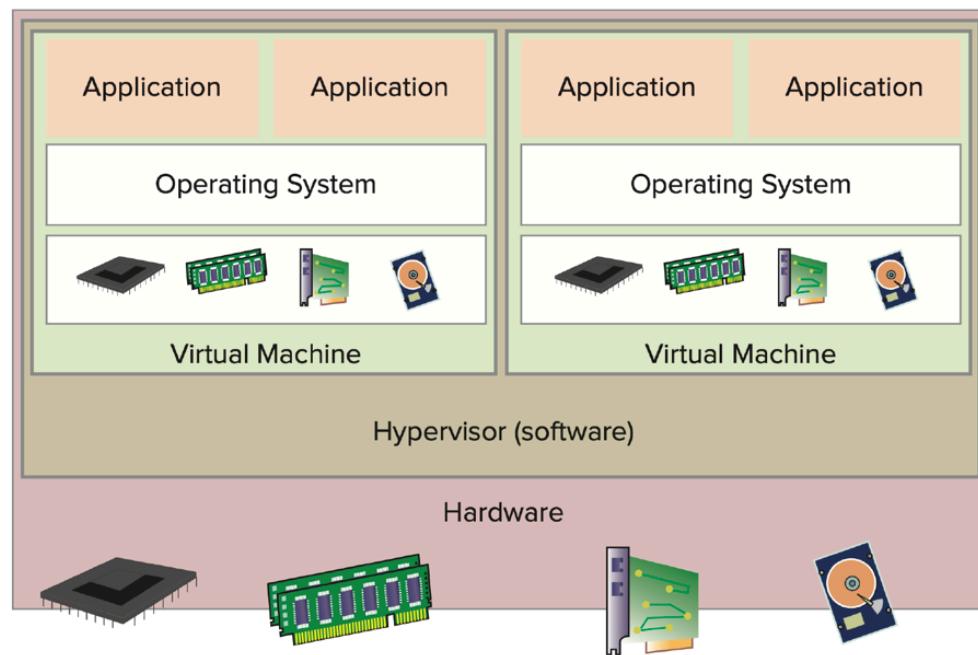


LES HYPERVISEURS

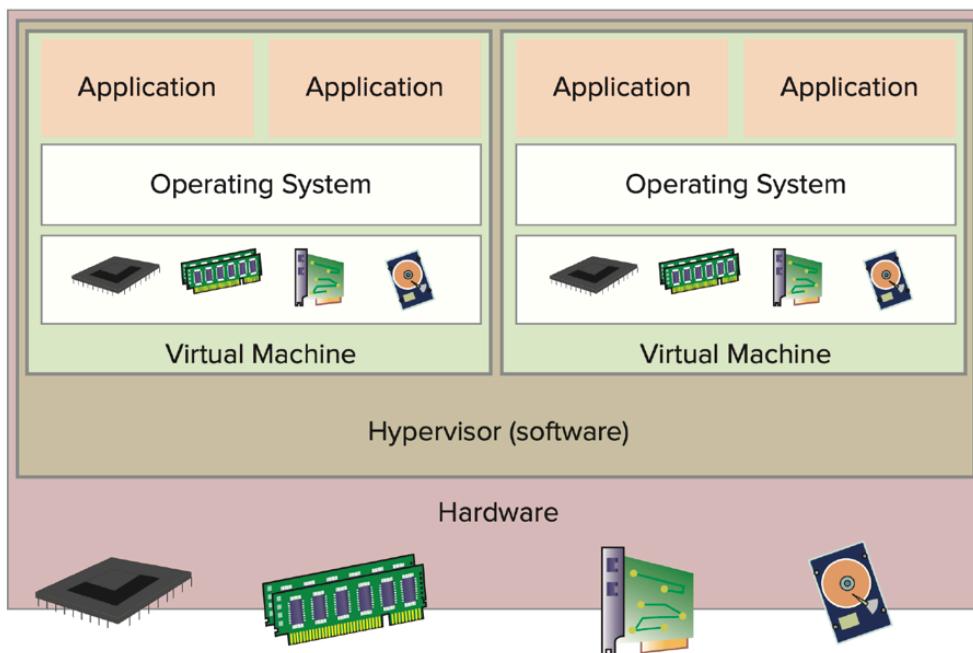


- Un hyperviseur est un composant logiciel qui gère les interactions entre les machines virtuelles et le matériel sous-jacent.

LE RÔLE DES HYPERVISEURS

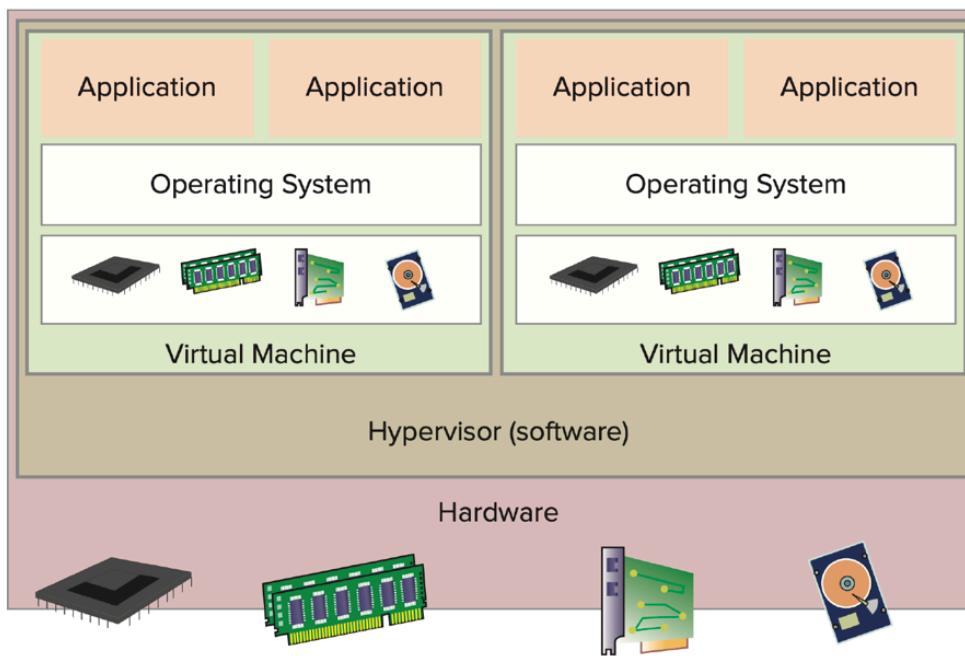


LE RÔLE DES HYPERVISEURS



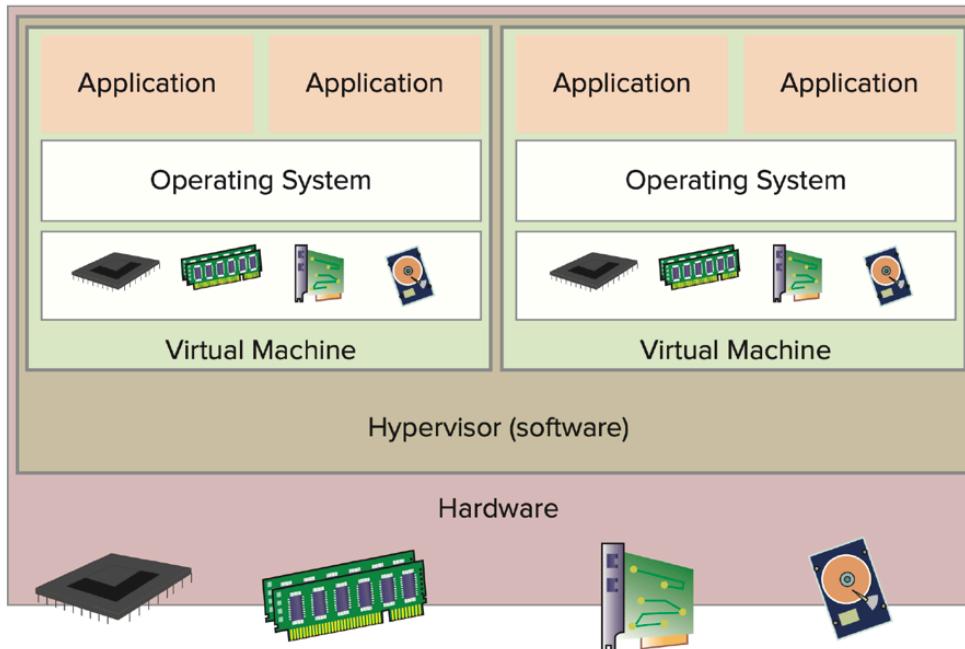
- Autorisez les VM à partager les mêmes ressources physiques.

LE RÔLE DES HYPERVISEURS



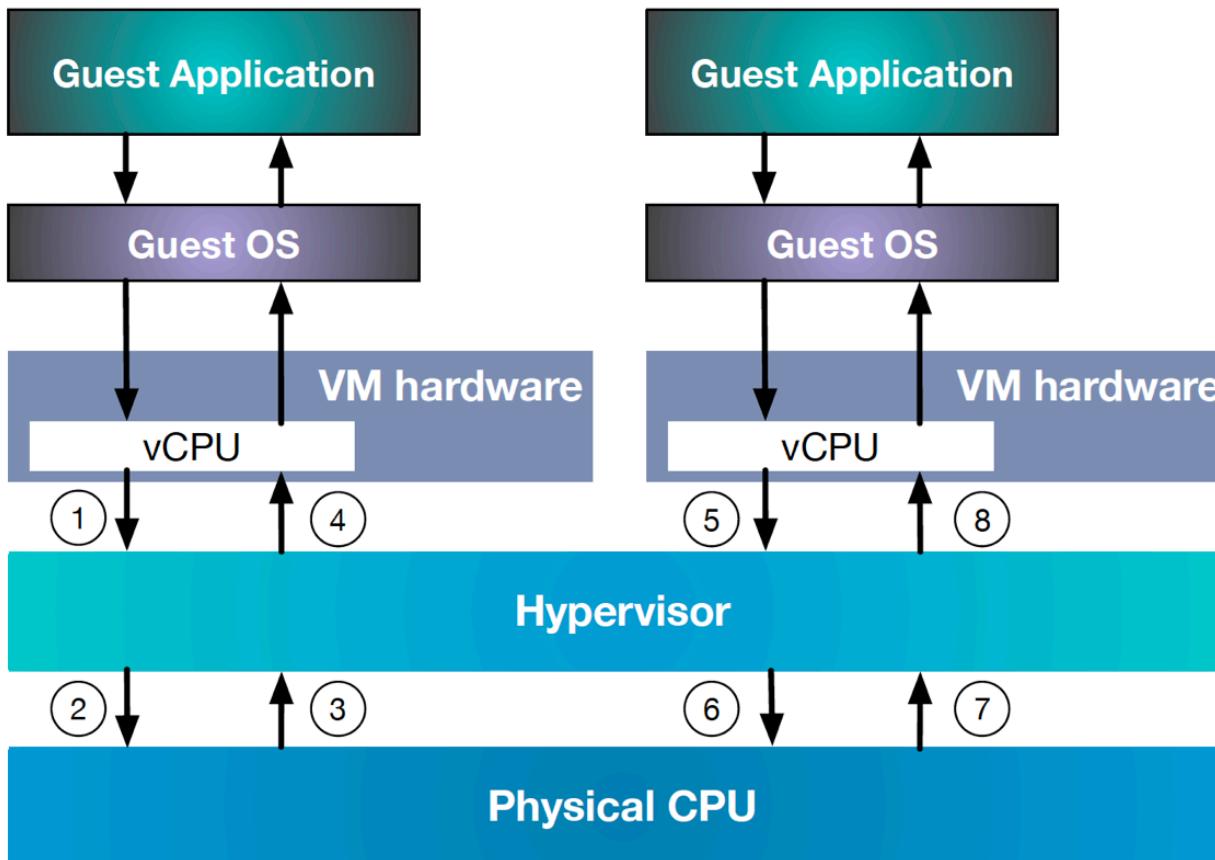
- Autorisez les **VM** à **partager les mêmes ressources** physiques.
- Présentez à chaque **VM une fraction** des ressources physiques.

LE RÔLE DES HYPERVISEURS

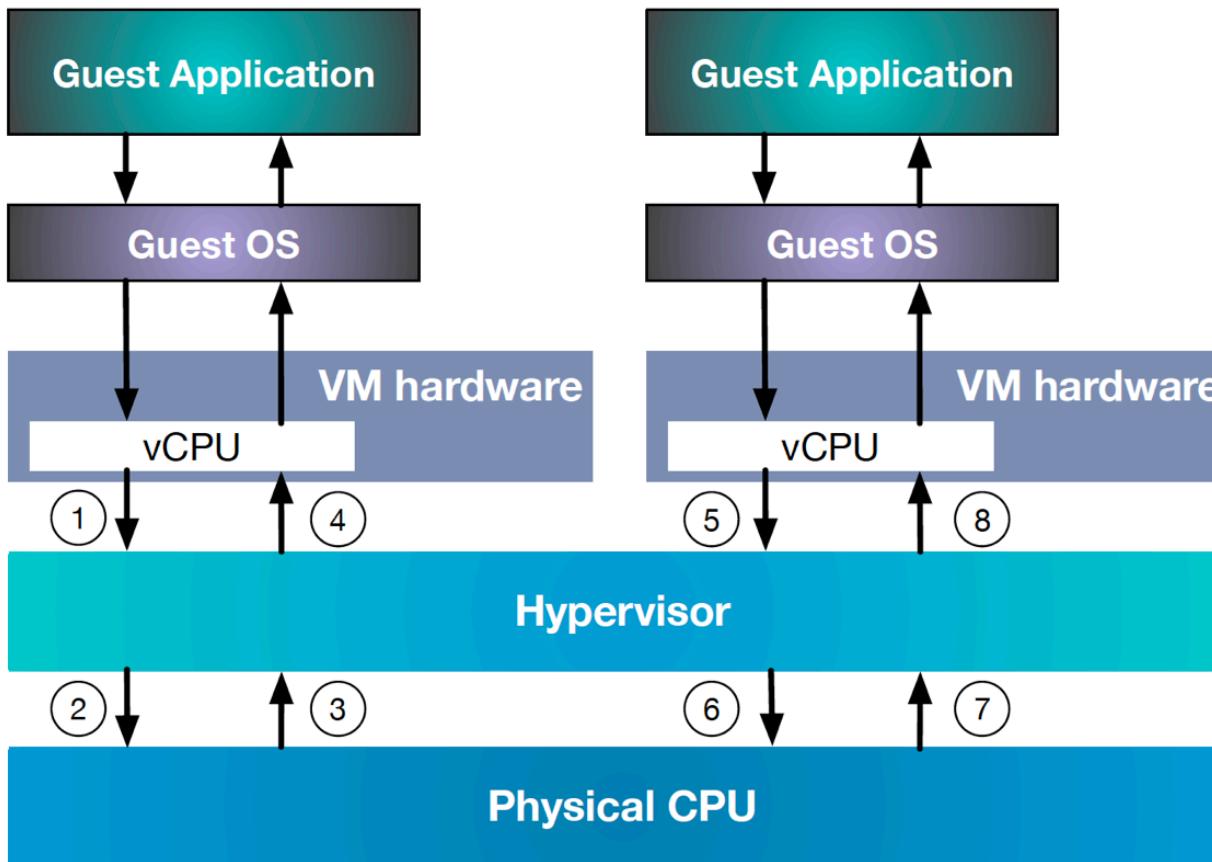


- Autorisez les **VM** à **partager les mêmes ressources** physiques.
- Présentez à chaque **VM une fraction** des ressources physiques.
- **Répond aux demandes** des **VM** invitées.

COMMENT LE CPU EST VIRTUALISÉ



COMMENT LE CPU EST VIRTUALISÉ



L'**hyperviseur** planifie des tranches de temps sur la **CPU physique** pour exécuter les instructions des **CPU virtuelles**.

COMMENT LE CPU EST VIRTUALISÉ

COMMENT LE CPU EST VIRTUALISÉ

- Les serveurs : **plusieurs processeurs** avec **plusieurs cœurs**.

COMMENT LE CPU EST VIRTUALISÉ

- Les serveurs : **plusieurs processeurs** avec **plusieurs cœurs**.
- Chaque **VM** peut avoir plusieurs **vCPU**.

COMMENT LE CPU EST VIRTUALISÉ

- Les serveurs : **plusieurs processeurs** avec **plusieurs cœurs**.
- Chaque **VM** peut avoir plusieurs **vCPU**.
- L'**hyperviseur** ne mappe pas statiquement **vCPU** à un cœur.

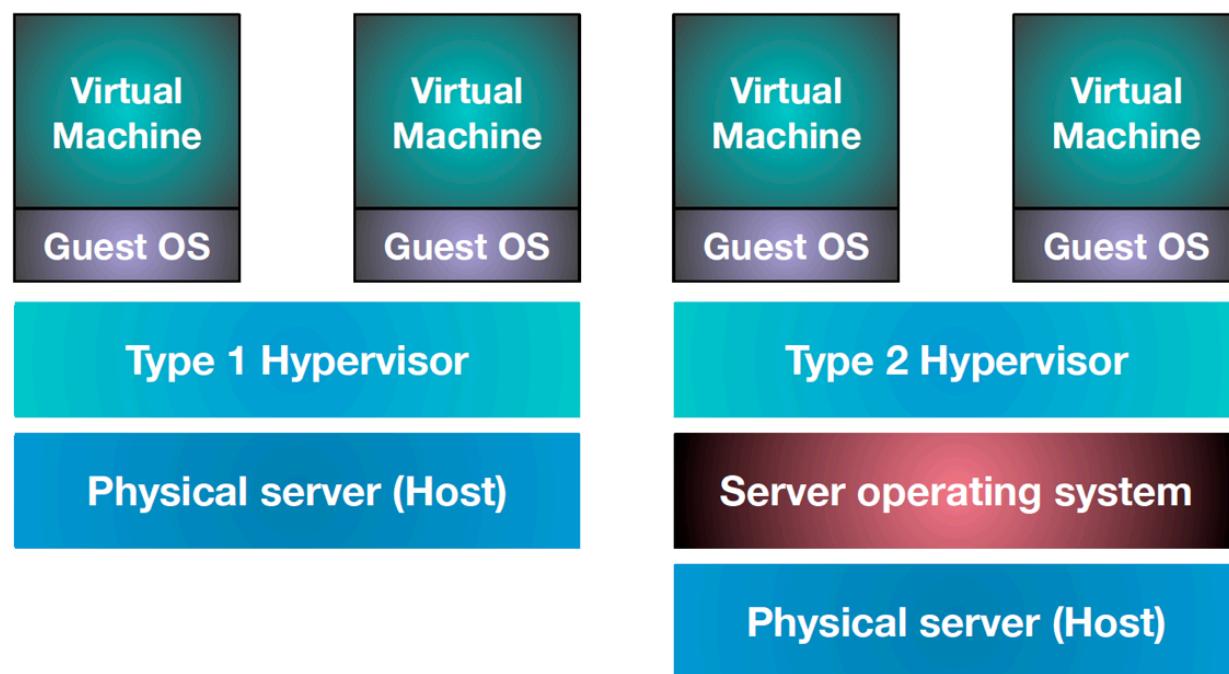
COMMENT LE CPU EST VIRTUALISÉ

- Les serveurs : **plusieurs processeurs** avec **plusieurs cœurs**.
- Chaque **VM** peut avoir plusieurs **vCPU**.
- L'**hyperviseur** ne mappe pas statiquement **vCPU** à un cœur.
👉 une **VM** ne peut pas avoir plus de **vCPU** que le nombre total de cœurs physiques.

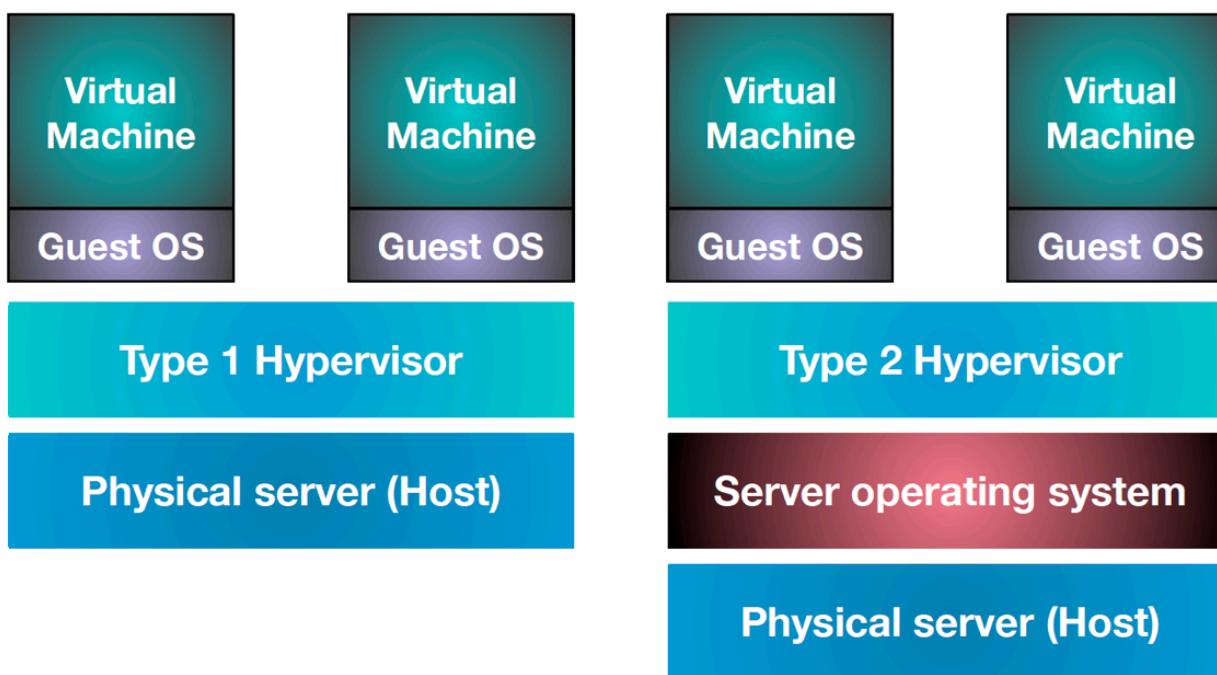
COMMENT LE CPU EST VIRTUALISÉ

- Les serveurs : **plusieurs processeurs** avec **plusieurs cœurs**.
- Chaque **VM** peut avoir plusieurs **vCPU**.
- L'**hyperviseur** ne mappe pas statiquement **vCPU** à un cœur.
 - 👉 une **VM** ne peut pas avoir plus de **vCPU** que le nombre total de cœurs physiques.
 - 👉 le nombre de **vCPU** sur toutes les **VM** peut dépasser le nombre total de cœurs physiques.

LES TYPES DE HYPERVISEURS

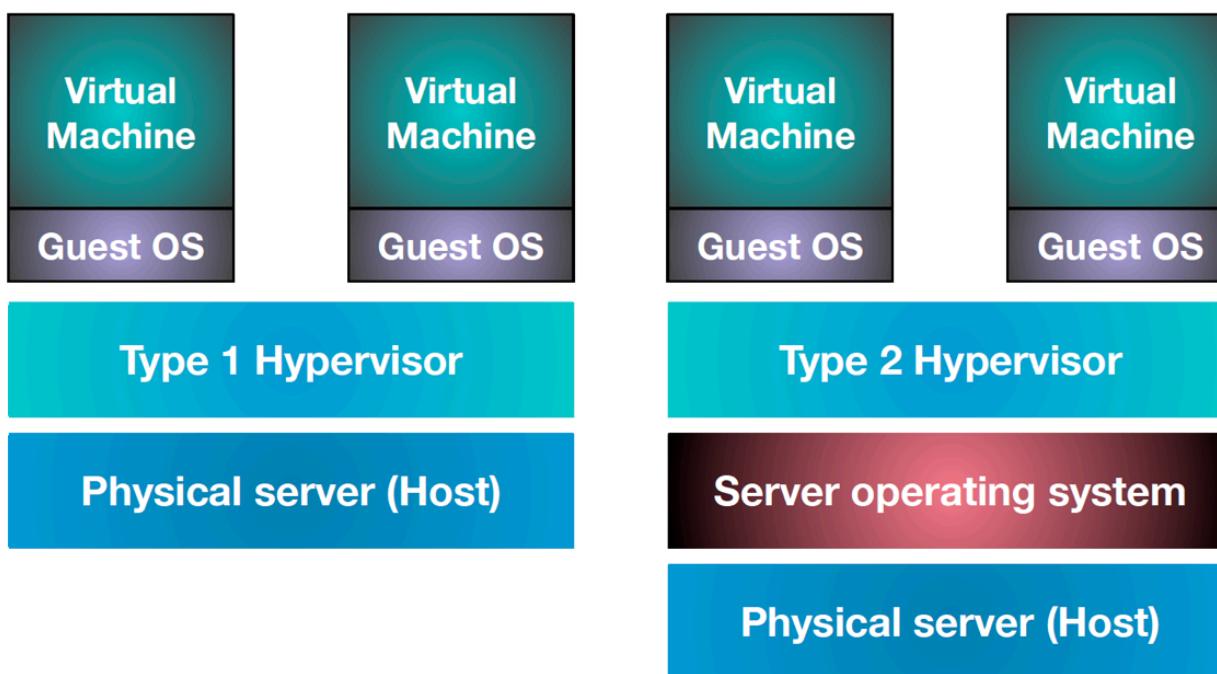


LES TYPES DE HYPERVISEURS



- **Type 1** : s'exécute sur le matériel hôte ([Microsoft Hyper-V](#))

LES TYPES DE HYPERVISEURS



- **Type 1** : s'exécute sur le matériel hôte ([Microsoft Hyper-V](#))
- **Type 2** : s'exécute en tant qu'application sur l'OS hôte ([VirtualBox](#))

LES AVANTAGES DES VMS

LES AVANTAGES DES VMS

- ✓ Plusieurs serveurs virtuels dans un serveur physique.

LES AVANTAGES DES VMS

- ✓ Plusieurs serveurs virtuels dans un serveur physique.
 - meilleure utilisation des ressources.

LES AVANTAGES DES VMS

- ✓ Plusieurs serveurs virtuels dans un serveur physique.
 - meilleure utilisation des ressources.
 - coûts inférieurs.

LES AVANTAGES DES VMS

- ✓ Plusieurs serveurs virtuels dans un serveur physique.
 - meilleure utilisation des ressources.
 - coûts inférieurs.
 - gestion simplifiée du data-center.

LES AVANTAGES DES VMS

- ✓ Plusieurs serveurs virtuels dans un serveur physique.
 - meilleure utilisation des ressources.
 - coûts inférieurs.
 - gestion simplifiée du data-center.

- ✓ Création facile de **VMs** à partir de **modèles/clones**.

LES AVANTAGES DES VMS

✓ Plusieurs serveurs virtuels dans un serveur physique.

- meilleure utilisation des ressources.
- coûts inférieurs.
- gestion simplifiée du data-center.

✓ Création facile de **VMs** à partir de **modèles/clones**.

- un serveur virtuel peut être opérationnel en quelques minutes/heures.

LES AVANTAGES DES VMS

✓ Plusieurs serveurs virtuels dans un serveur physique.

- meilleure utilisation des ressources.
- coûts inférieurs.
- gestion simplifiée du data-center.

✓ Création facile de VMs à partir de modèles/clones.

- un serveur virtuel peut être opérationnel en quelques minutes/heures.
- 👉 un serveur physique peut avoir besoin de semaines.

LES AVANTAGES DES VMS

✓ Plusieurs serveurs virtuels dans un serveur physique.

- meilleure utilisation des ressources.
- coûts inférieurs.
- gestion simplifiée du data-center.

✓ Création facile de **VMs** à partir de **modèles/clones**.

- un serveur virtuel peut être opérationnel en quelques minutes/heures.
- 👉 un serveur physique peut avoir besoin de semaines.

✓ Migration facile des **VMs**.

LES AVANTAGES DES VMS

✓ Plusieurs serveurs virtuels dans un serveur physique.

- meilleure utilisation des ressources.
- coûts inférieurs.
- gestion simplifiée du data-center.

✓ Création facile de **VMs** à partir de **modèles/clones**.

- un serveur virtuel peut être opérationnel en quelques minutes/heures.
- 👉 un serveur physique peut avoir besoin de semaines.

✓ Migration facile des **VMs**.

- temps d'arrêt réduits ou minimisés.

LES CONTENEURS

⌚ Niveaux de virtualisation

POURQUOI LA CONTENEURISATION ?

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

- Machines virtuelles

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

- **Machines virtuelles**
 - une bonne solution pour **packager un microservice.**

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

- **Machines virtuelles**
 - une bonne solution pour **packager un microservice**.
 - le microservice est **intégré** à la machine virtuelle.

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

- **Machines virtuelles**
 - une bonne solution pour **packager un microservice**.
 - le microservice est **intégré** à la machine virtuelle.
 - **déploiement facile** sur n'importe quelle machine.

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

- **Machines virtuelles**
 - une bonne solution pour **packager un microservice**.
 - le microservice est **intégré** à la machine virtuelle.
 - **déploiement facile** sur n'importe quelle machine.
- ... cependant, les **VMs sont lourdes** :

POURQUOI LA CONTENEURISATION ?

Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

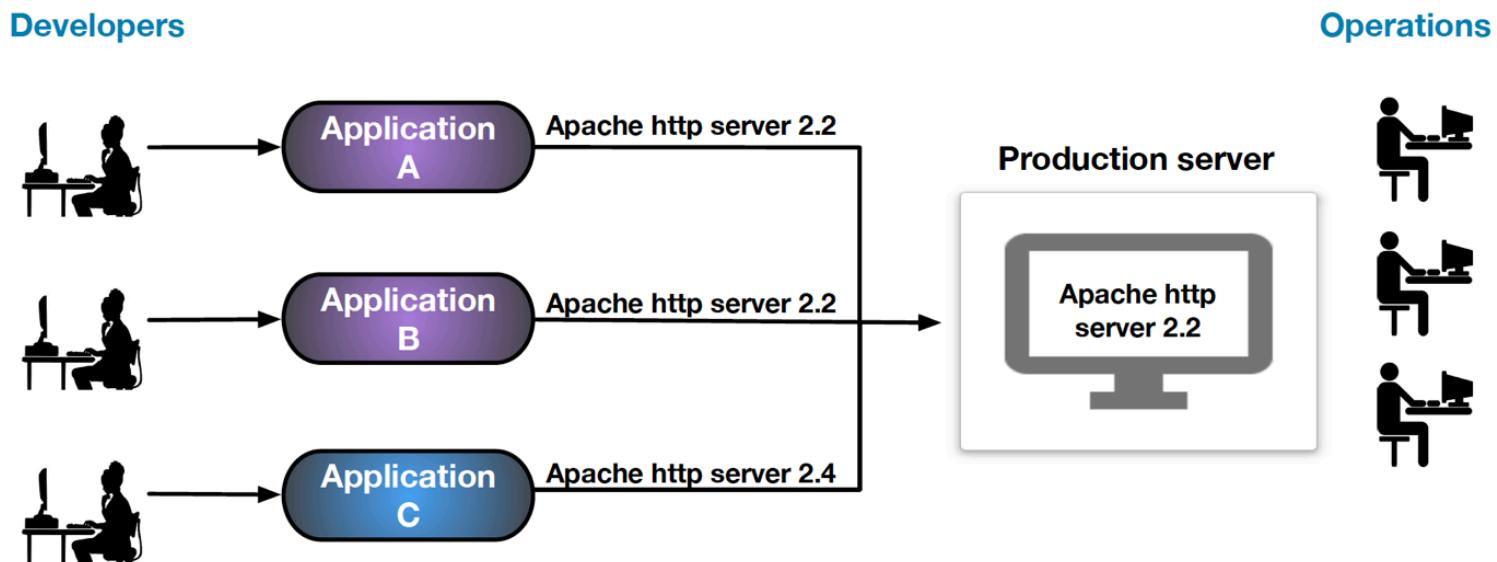
- **Machines virtuelles**
 - une bonne solution pour **packager un microservice**.
 - le microservice est **intégré** à la machine virtuelle.
 - **déploiement facile** sur n'importe quelle machine.
- ... cependant, les **VMs sont lourdes** :
 - ✖ elles contiennent un **OS complet**.

POURQUOI LA CONTENEURISATION ?

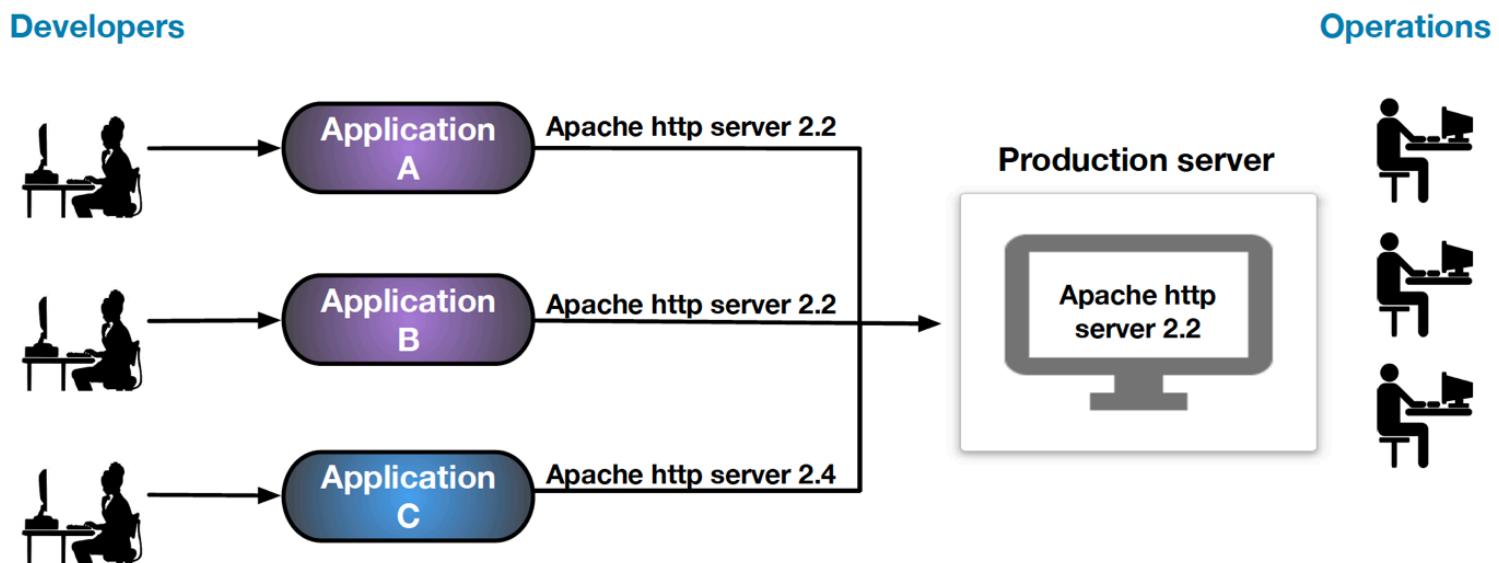
Gabriel N. Schenker : Utiliser une VM pour packager une seule application revient à utiliser « *un bateau gigantesque pour transporter un camion chargé de bananes* ».

- **Machines virtuelles**
 - une bonne solution pour **packager un microservice**.
 - le microservice est **intégré** à la machine virtuelle.
 - **déploiement facile** sur n'importe quelle machine.
- ... cependant, les **VMs sont lourdes** :
 - ✗ elles contiennent un **OS complet**.
 - ✗ le **démarrage** ou la **migration** d'une VM peut prendre du temps.

POURQUOI LA CONTENEURISATION ?

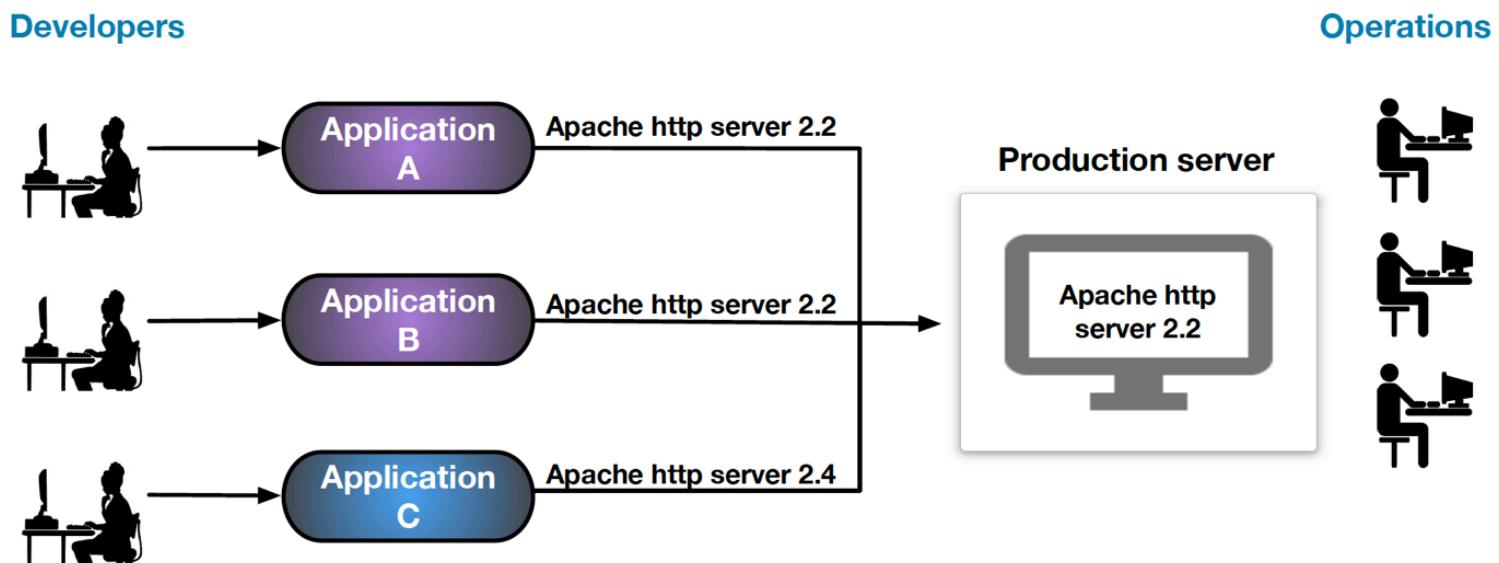


POURQUOI LA CONTENEURISATION ?



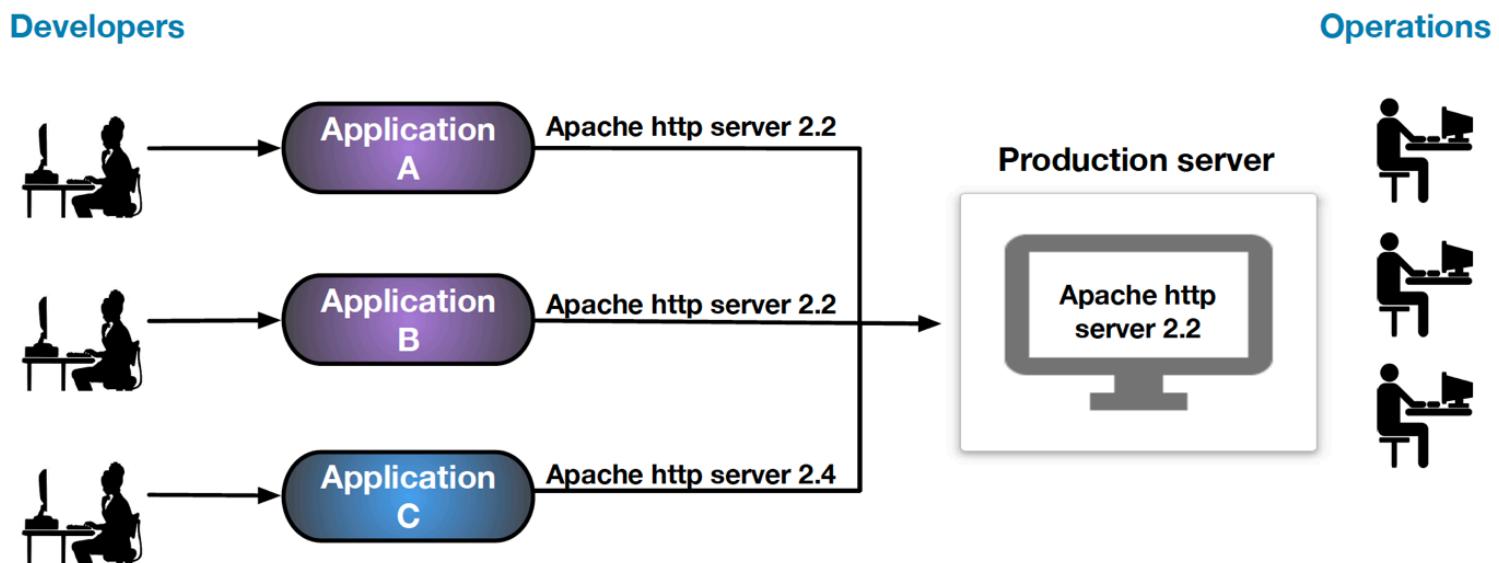
- ✗ Chaque **application** est livrée avec un "**package différent**".

POURQUOI LA CONTENEURISATION ?



- ✗ Chaque **application** est livrée avec un "**package différent**".
- ✗ **Déploiement** dépend de l'application.

POURQUOI LA CONTENEURISATION ?



- ✖ Chaque **application** est livrée avec un "**package différent**".
- ✖ **Déploiement** dépend de l'application.
- ✖ Gestion de **dépendance** difficile en production.

POURQUOI LA CONTENEURISATION ?

Solution

POURQUOI LA CONTENEURISATION ?

Solution

- Les conteneurs standardisent le développement et le déploiement d'applications.

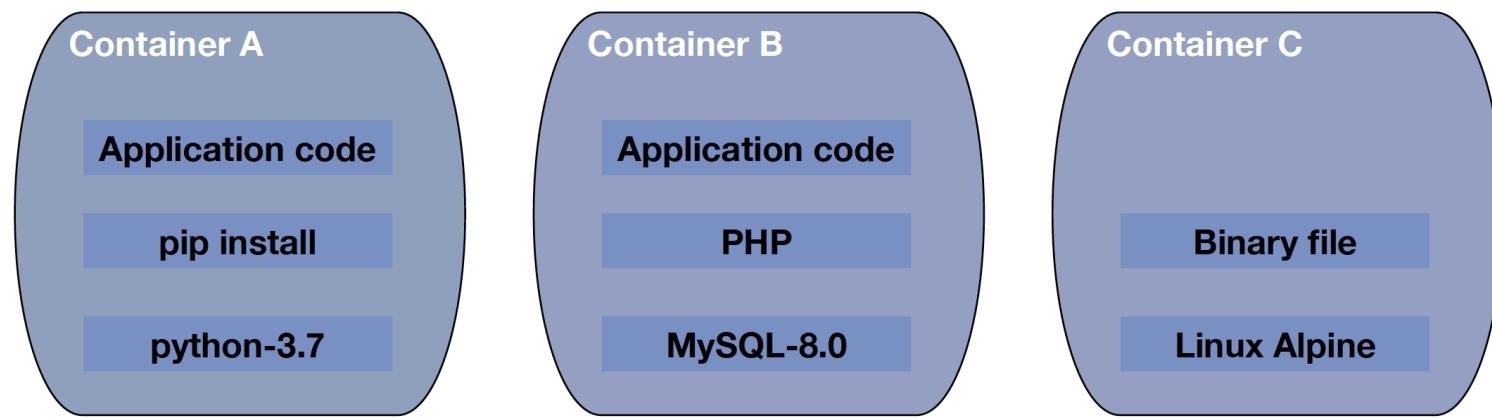
POURQUOI LA CONTENEURISATION ?

Solution

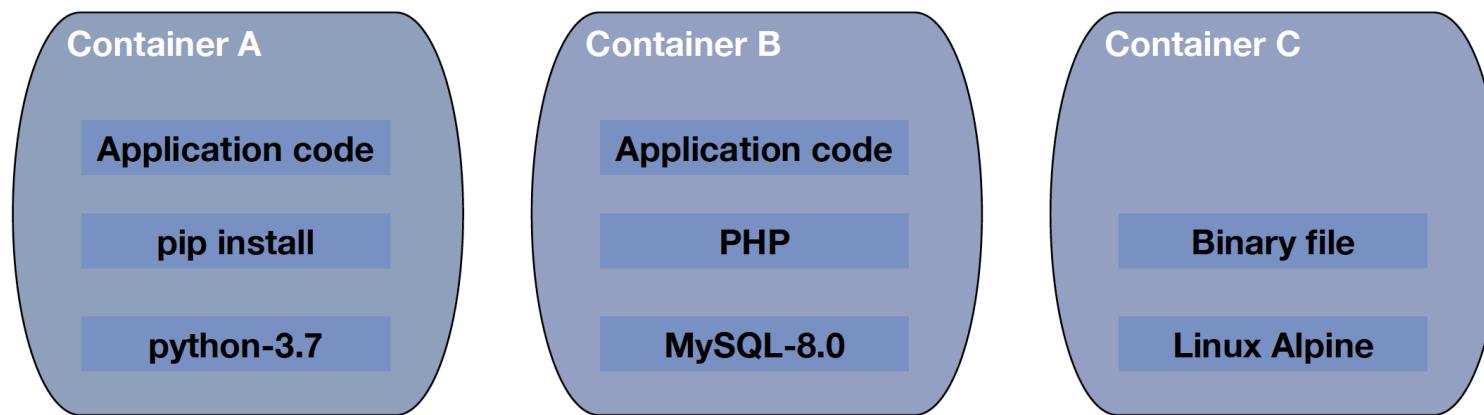
- Les conteneurs standardisent le développement et le déploiement d'applications.
- Une excellente solution dans un environnement cloud.

C'EST QUOI UN CONTENEUR ?

C'EST QUOI UN CONTENEUR ?



C'EST QUOI UN CONTENEUR ?

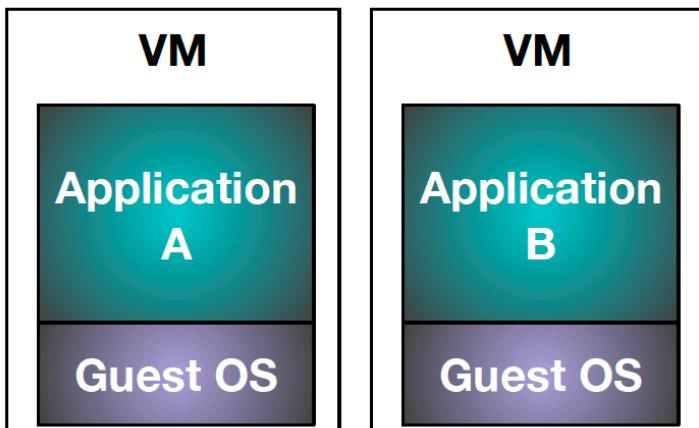


Une unité logicielle qui regroupe le **code** et **toutes ses dépendances** afin que l'application **s'exécute** rapidement et de manière fiable **d'un environnement à un autre**.

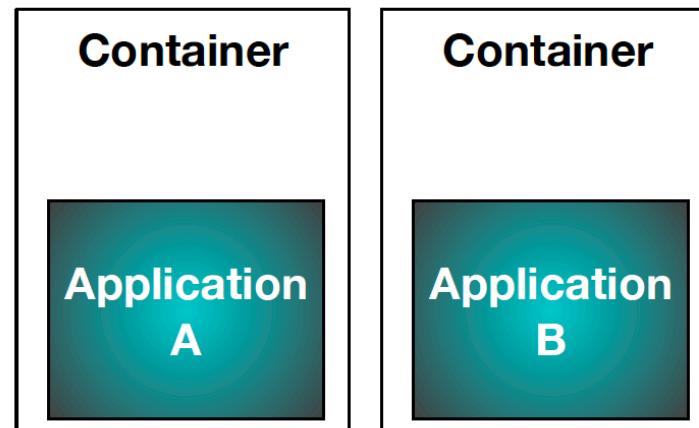
VM vs CONTENEUR

VM vs CONTENEUR

Virtual machines



Containers

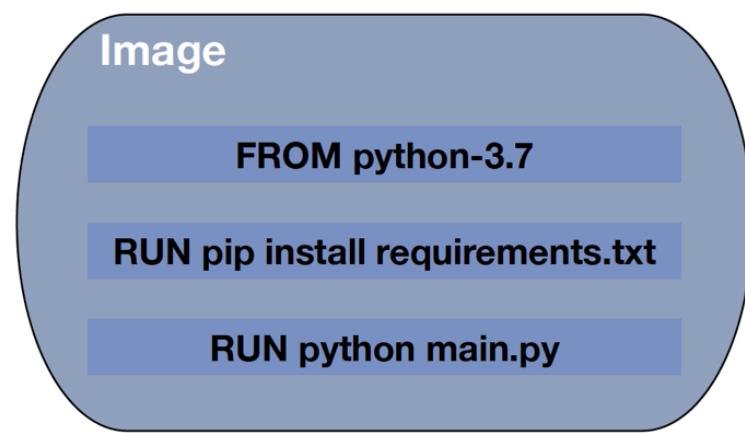


Container manager

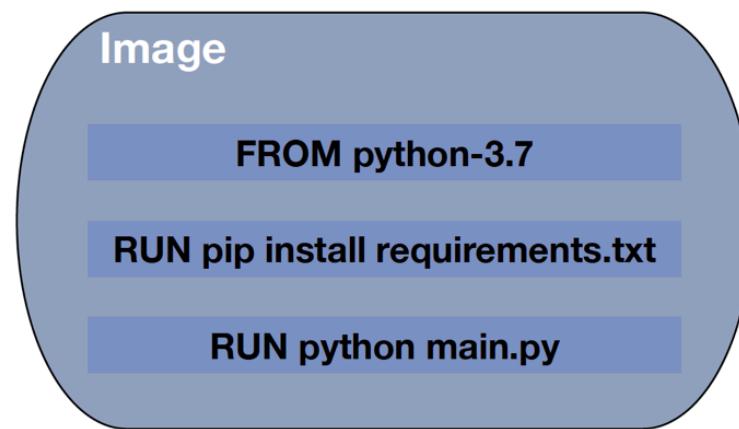
Host operating system

Hardware

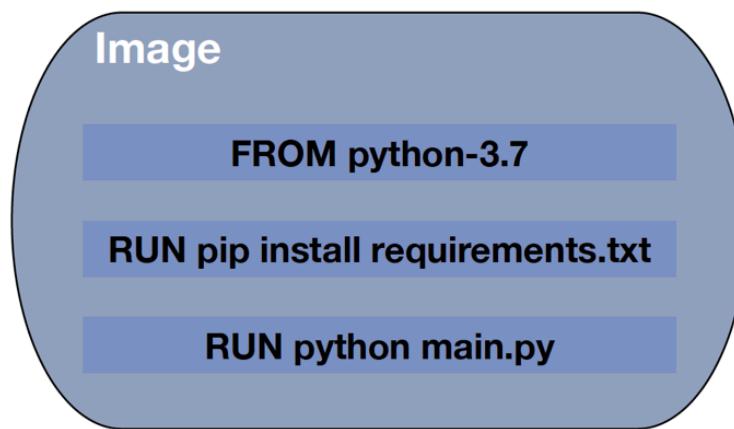
DOCKER



DOCKER

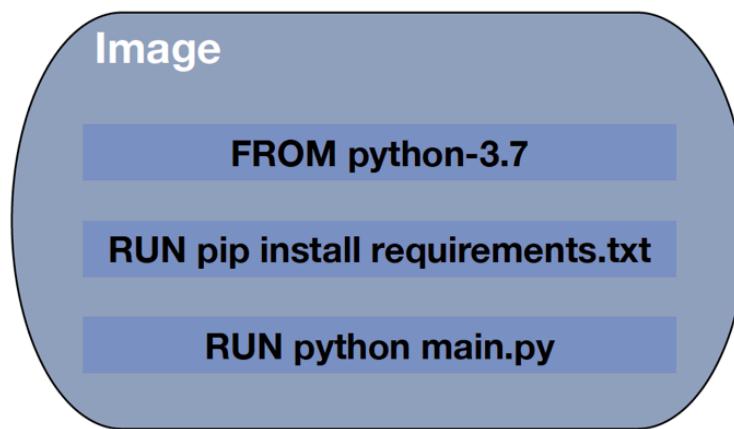


DOCKER



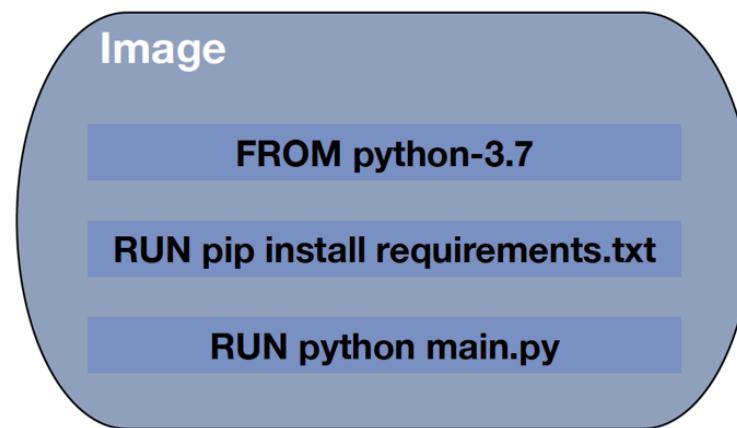
- Une plate-forme ouverte pour le développement, la livraison et l'exécution d'applications.

DOCKER



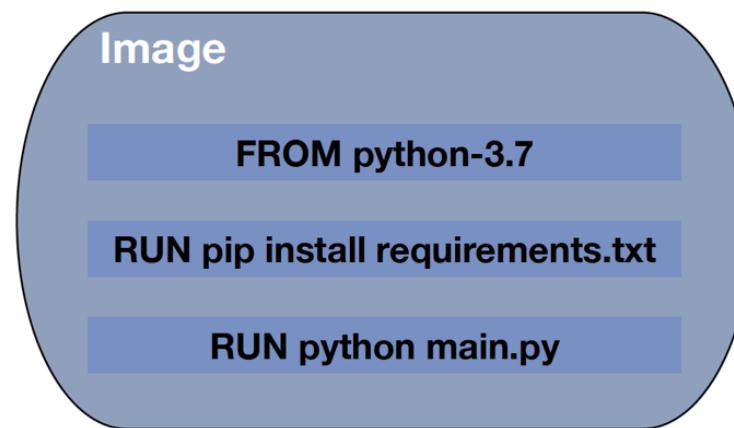
- Une plate-forme ouverte pour le développement, la livraison et l'exécution d'applications.
- Il offre la possibilité de packager et d'exécuter une application dans un environnement isolé appelé conteneur

DOCKER



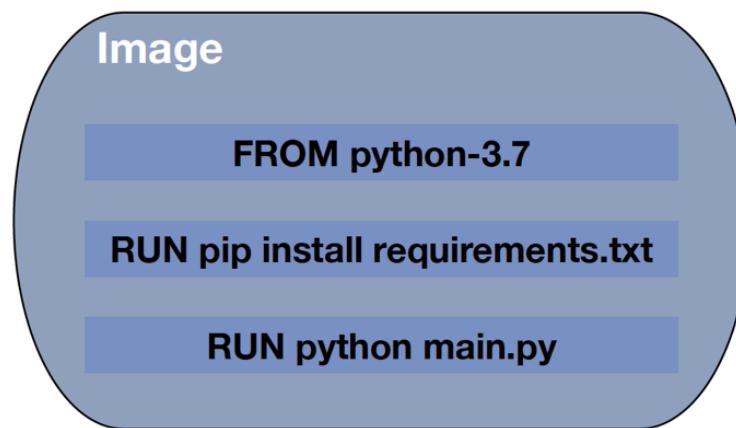
- **Docker** crée et exécute **un conteneur** à partir d'une **image**.

DOCKER



- Docker crée et exécute un conteneur à partir d'une image.
- Image : modèle en lecture seule avec des instructions pour créer et exécuter un conteneur.

DOCKER

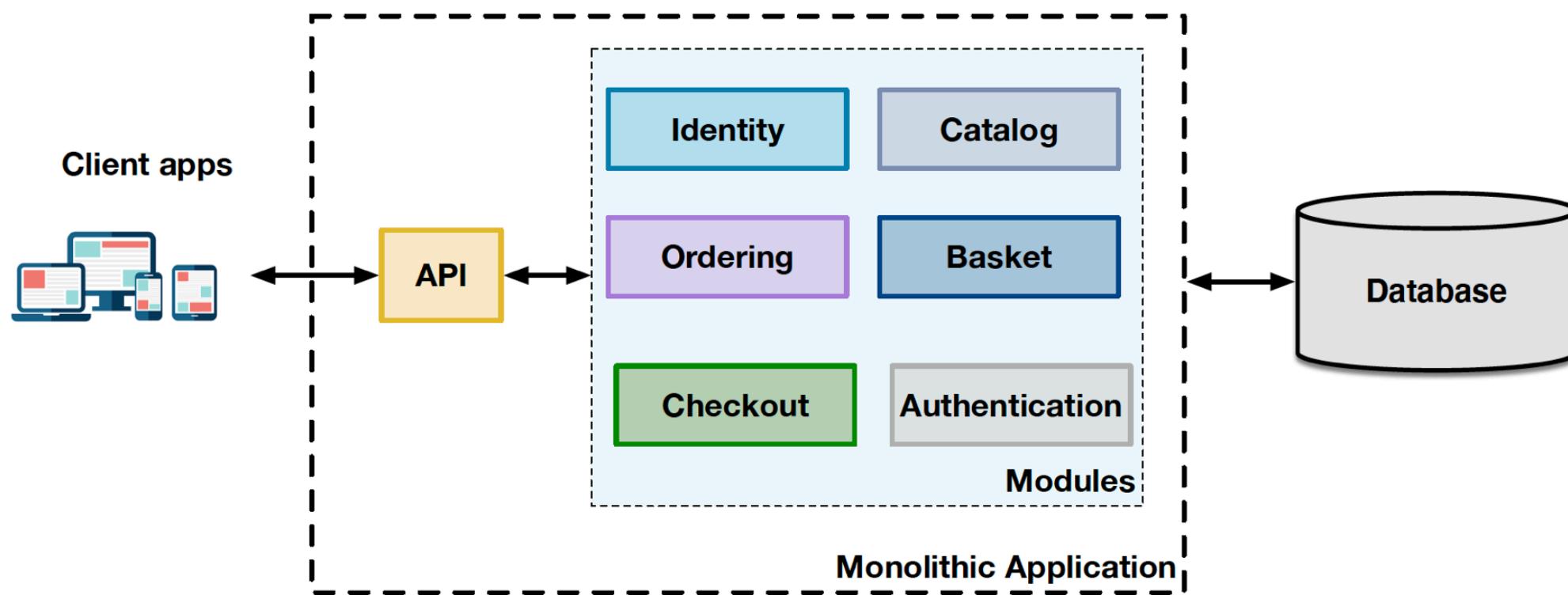


- Docker crée et exécute un conteneur à partir d'une image.
- Image : modèle en lecture seule avec des instructions pour créer et exécuter un conteneur.
- Image contient tous les fichiers nécessaires pour exécuter une application dans un conteneur.

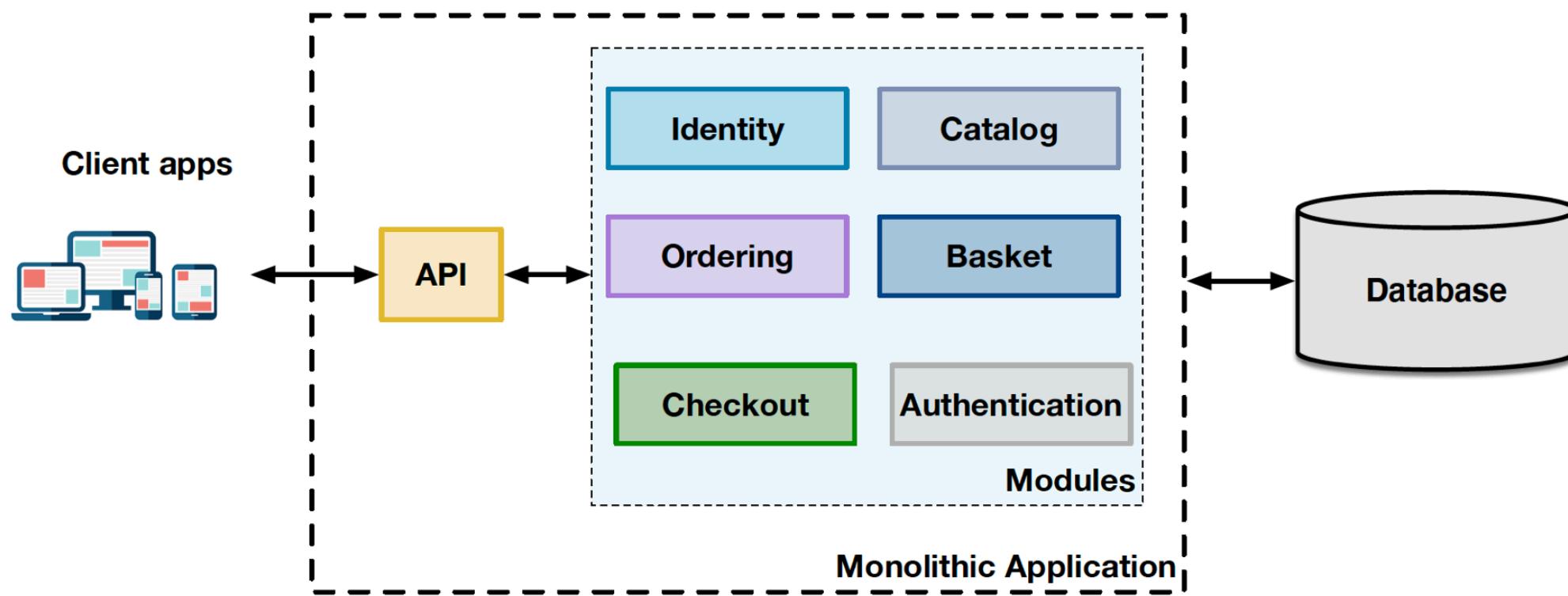
APPLICATIONS CLOUD NATIVE

⌚ Niveaux de virtualisation

APPLICATIONS TRADITIONNELLES

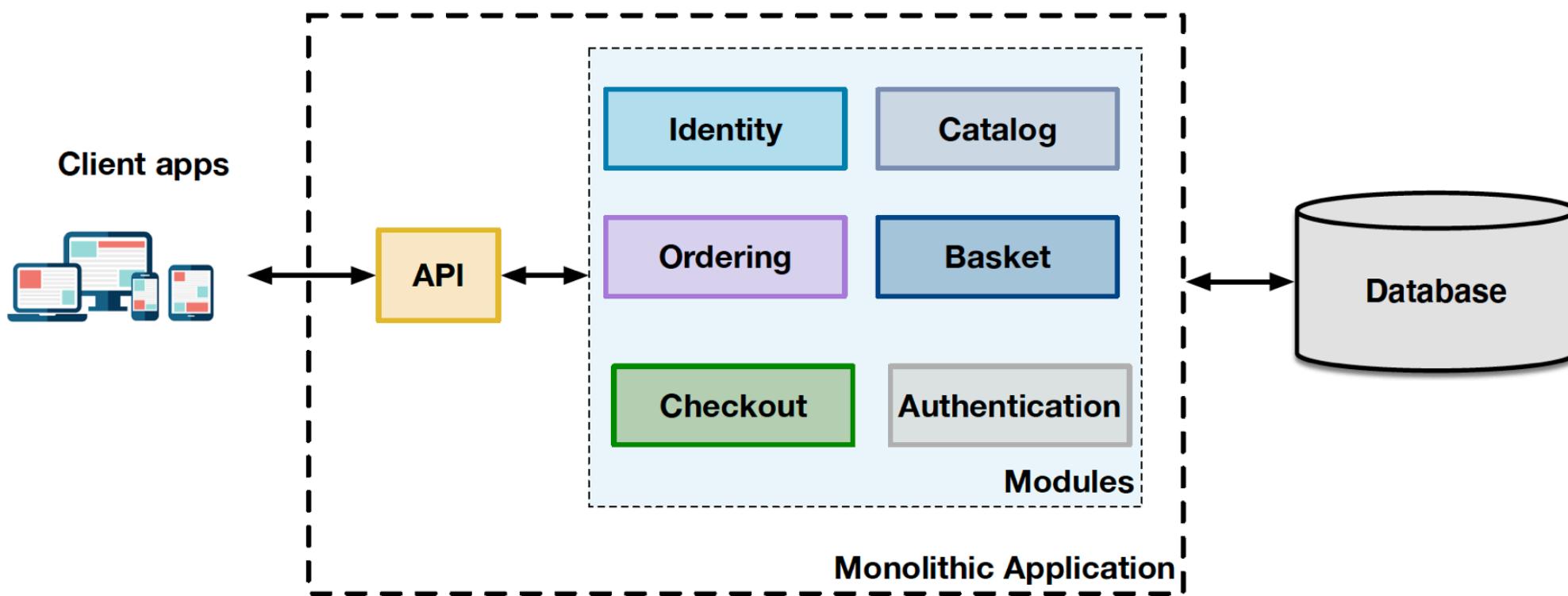


APPLICATIONS TRADITIONNELLES



- Les applications traditionnelles ont une architecture monolithique.

APPLICATIONS TRADITIONNELLES



- Les applications traditionnelles ont une architecture monolithique.
- Toute la logique métier est intégrée dans une seule application.

LES INCONVÉNIENTS

LES INCONVÉNIENTS

- ✗ Le **code** est difficile à maintenir.

LES INCONVÉNIENTS

- ✗ Le **code** est difficile à maintenir.
- ✗ La **base de données** est difficile à maintenir.

LES INCONVÉNIENTS

- ✗ Le **code** est difficile à maintenir.
- ✗ La **base de données** est difficile à maintenir.
- ✗ Chaque **modification** entraîne le **déploiement de l'ensemble de l'application**.

LES INCONVÉNIENTS

- ✗ Le **code** est difficile à maintenir.
- ✗ La **base de données** est difficile à maintenir.
- ✗ Chaque **modification** entraîne le **déploiement de l'ensemble de l'application**.
- ✗ Une **erreur** dans un module affecte **l'ensemble de l'application**.

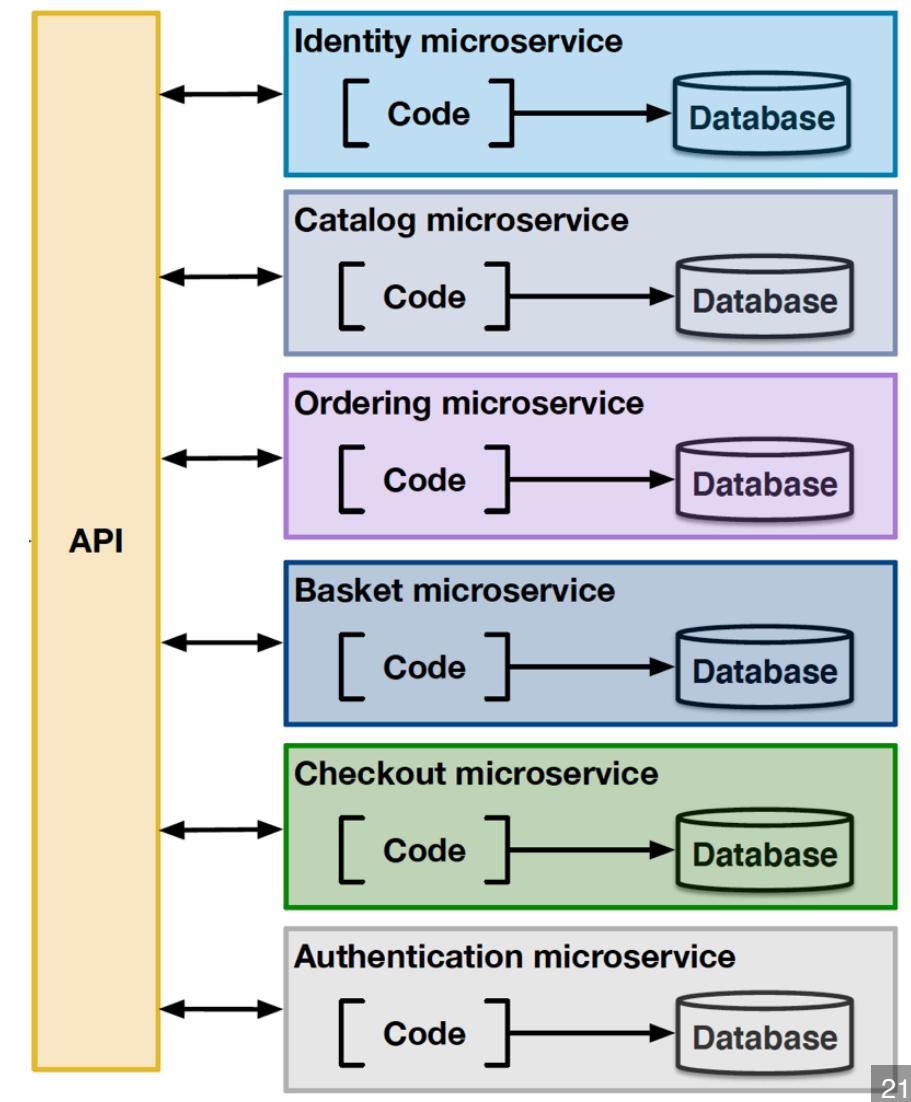
LES INCONVÉNIENTS

- ✗ Le **code** est difficile à maintenir.
- ✗ La **base de données** est difficile à maintenir.
- ✗ Chaque **modification** entraîne le **déploiement de l'ensemble de l'application**.
- ✗ Une **erreur** dans un module affecte **l'ensemble de l'application**.
- ✗ **Difficile** d'adhérer aux **pratiques DevOps**.

LES INCONVÉNIENTS

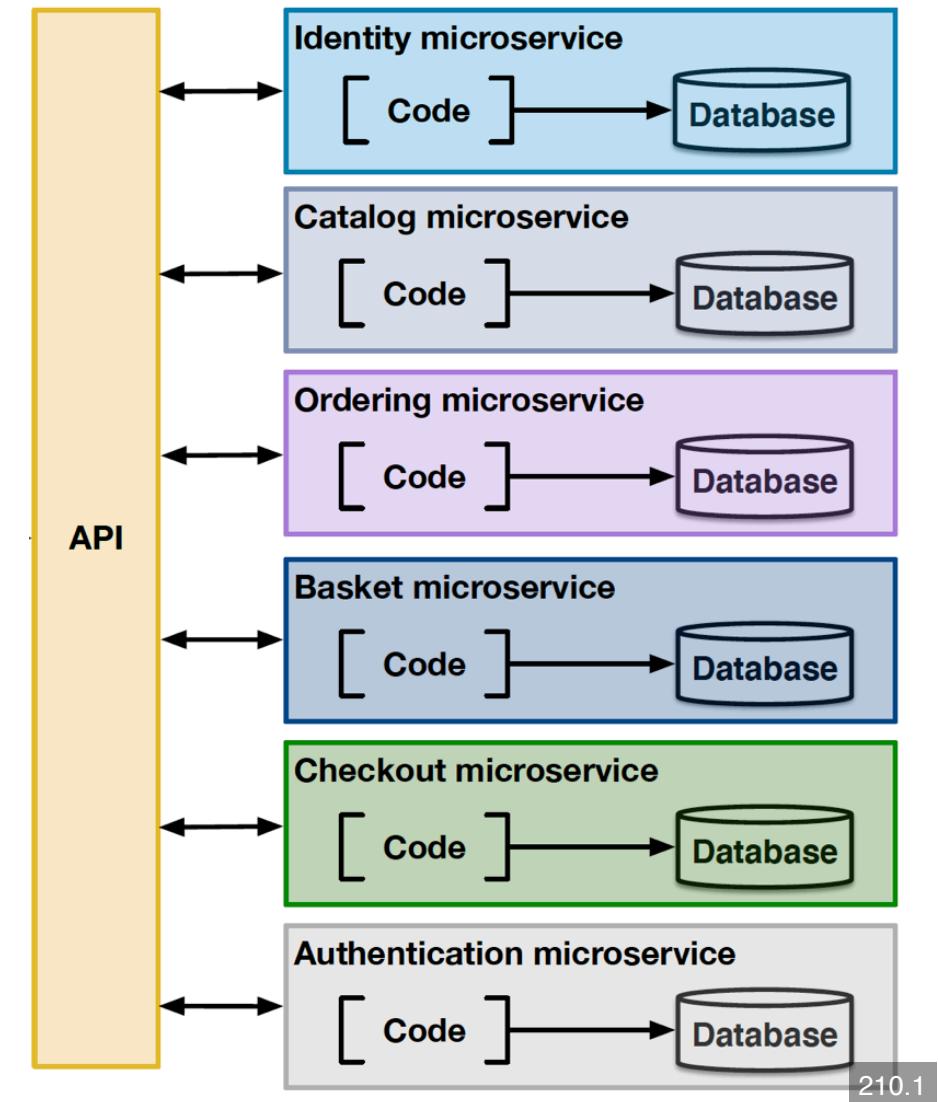
- ✗ Le **code** est difficile à maintenir.
- ✗ La **base de données** est difficile à maintenir.
- ✗ Chaque **modification** entraîne le **déploiement de l'ensemble de l'application**.
- ✗ Une **erreur** dans un module affecte **l'ensemble de l'application**.
- ✗ **Difficile** d'adhérer aux **pratiques DevOps**.
 - **DevOps** : modularité, livraison, déploiement continu, isolation des pannes ...

ARCHITECTURES MICROSERVICES



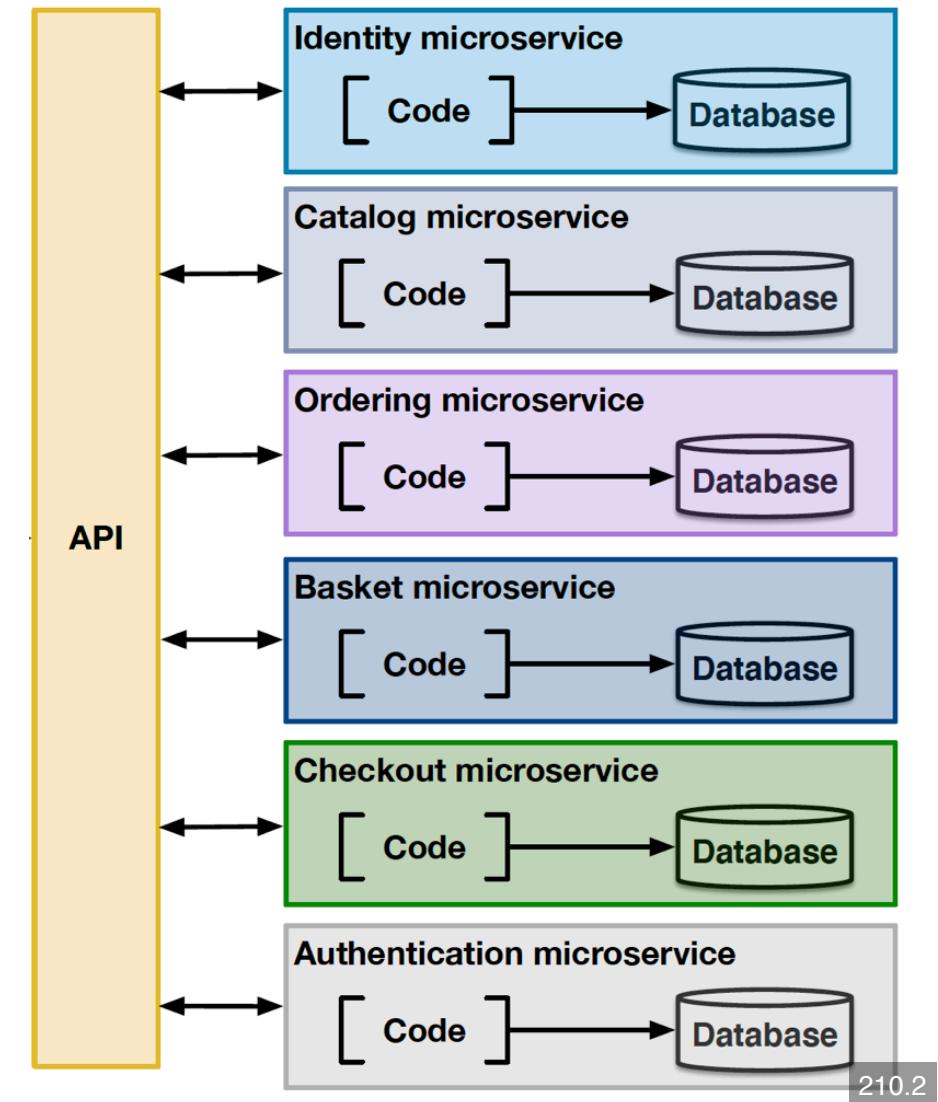
ARCHITECTURES MICROSERVICES

- Application : **microservices faiblement couplés**



ARCHITECTURES MICROSERVICES

- Application : **microservices** faiblement couplés
- Chaque **microservice** est un **processus indépendant**.



LES AVANTAGES

LES AVANTAGES

- ✓ Respect des principes DevOps.

LES AVANTAGES

- ✓ Respect des principes DevOps.
- ✓ Différents microservices, différentes technologies.

LES AVANTAGES

- ✓ Respect des principes **DevOps**.
- ✓ Différents microservices, **différentes technologies**.
- ✓ Microservices **évoluent indépendamment** les uns des autres.

LES AVANTAGES

- ✓ Respect des principes DevOps.
- ✓ Différents microservices, différentes technologies.
- ✓ Microservices évoluent indépendamment les uns des autres.
- ✓ BDD décentralisées :
technologie de BDD personnalisée pour chaque microservice.

LES AVANTAGES

- ✓ Respect des principes DevOps.
- ✓ Différents microservices, différentes technologies.
- ✓ Microservices évoluent indépendamment les uns des autres.
- ✓ BDD décentralisées :
technologie de BDD personnalisée pour chaque microservice.
- ✓ Bon choix architectural pour une application cloud native.

DÉPLOYS DES MICROSERVICES

DÉPLOYS DES MICROSERVICES

- Sans virtualisation

DÉPLOYS DES MICROSERVICES

- Sans virtualisation
 - plusieurs instances de service par **hôte**.
👉 chaque **hôte physique** exécute plusieurs instances de service.

DÉPLOYS DES MICROSERVICES

- **Sans virtualisation**

- plusieurs instances de service par **hôte**.
👉 chaque **hôte physique** exécute plusieurs instances de service.
- une seule instance de service par **hôte**.
👉 Chaque **hôte physique** exécute une seule instance de service.

DÉPLOYS DES MICROSERVICES

- **Sans virtualisation**
 - plusieurs instances de service par **hôte**.
👉 chaque **hôte physique** exécute plusieurs instances de service.
 - une seule instance de service par **hôte**.
👉 Chaque **hôte physique** exécute une seule instance de service.
- **Avec virtualisation**

DÉPLOYS DES MICROSERVICES

- **Sans virtualisation**
 - plusieurs instances de service par **hôte**.
 - 👉 chaque **hôte physique** exécute plusieurs instances de service.
 - une seule instance de service par **hôte**.
 - 👉 Chaque **hôte physique** exécute une seule instance de service.
- **Avec virtualisation**
 - une seule instance de service par **machine virtuelle (VM)**.
 - 👉 un service s'exécute dans une machine virtuelle.

DÉPLOYS DES MICROSERVICES

- **Sans virtualisation**

- plusieurs instances de service par **hôte**.
 - 👉 chaque **hôte physique** exécute plusieurs instances de service.
- une seule instance de service par **hôte**.
 - 👉 Chaque **hôte physique** exécute une seule instance de service.

- **Avec virtualisation**

- une seule instance de service par **machine virtuelle (VM)**.
 - 👉 **un service** s'exécute dans **une machine virtuelle**.
- une seule instance de service par **conteneur**.
 - 👉 **un service** s'exécute dans **un conteneur**.

DÉPLOYS DES MICROSERVICES

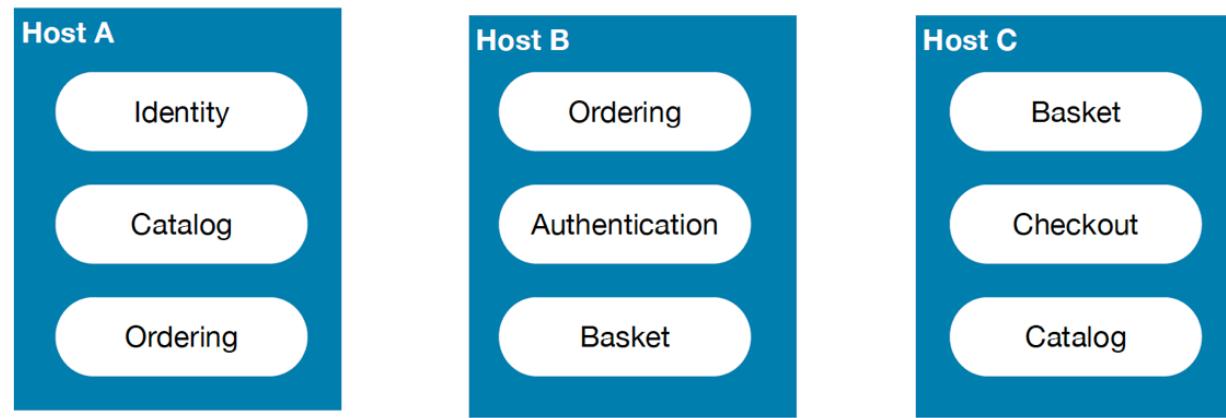
- **Sans virtualisation**

- plusieurs instances de service par **hôte**.
 - 👉 chaque **hôte physique** exécute plusieurs instances de service.
- une seule instance de service par **hôte**.
 - 👉 Chaque **hôte physique** exécute une seule instance de service.

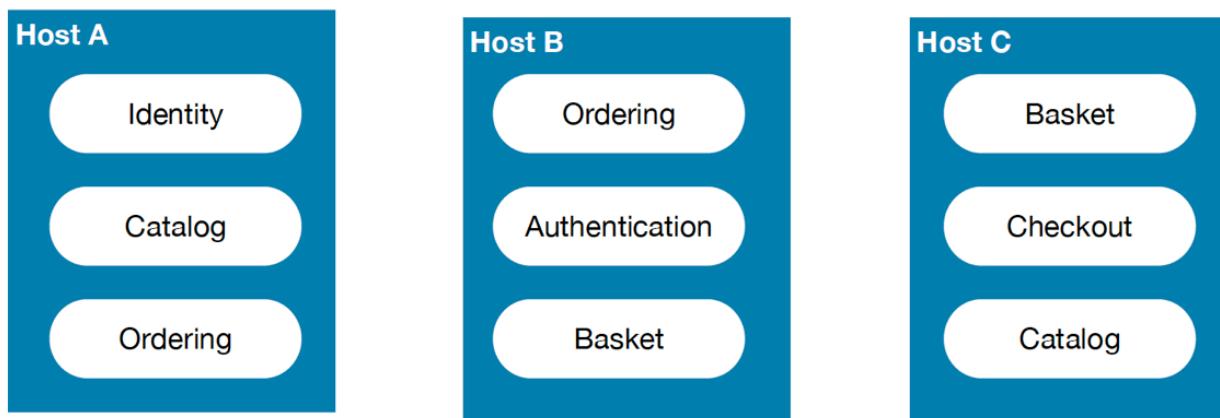
- **Avec virtualisation**

- une seule instance de service par **machine virtuelle (VM)**.
 - 👉 un **service** s'exécute dans une **machine virtuelle**.
- une seule instance de service par **conteneur**.
 - 👉 un **service** s'exécute dans un **conteneur**.
- de nombreuses **VM/conteneurs** s'exécutent sur **le même hôte physique**.

PLUSIEURS INSTANCES PAR HÔTE

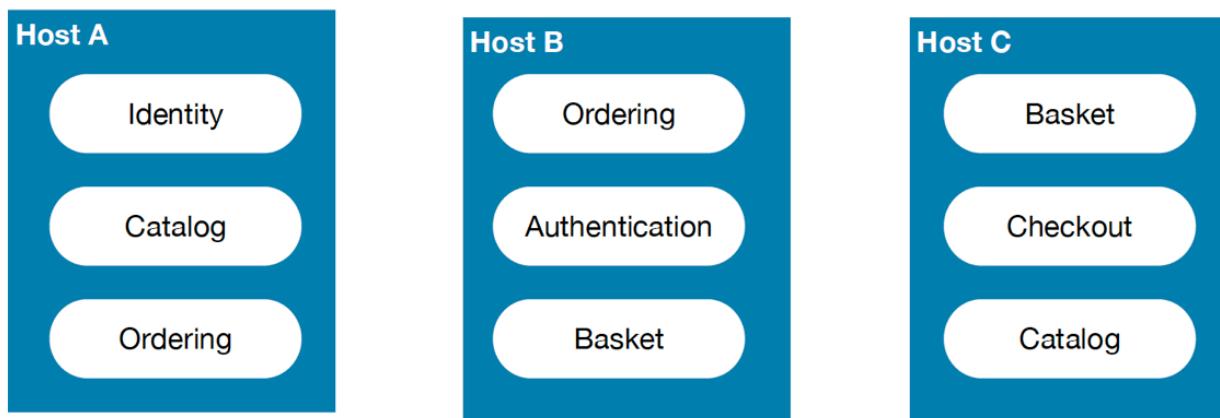


PLUSIEURS INSTANCES PAR HÔTE



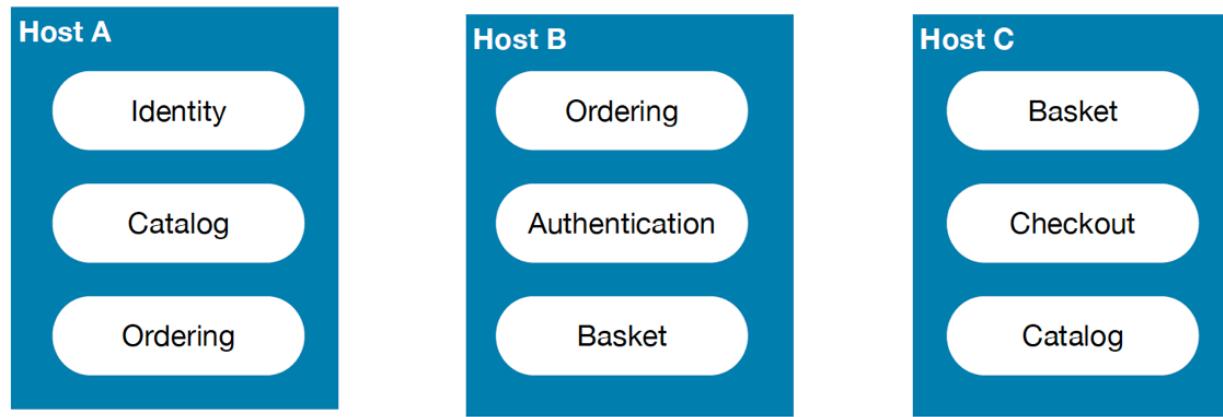
✓ Utilisation efficace des ressources.

PLUSIEURS INSTANCES PAR HÔTE



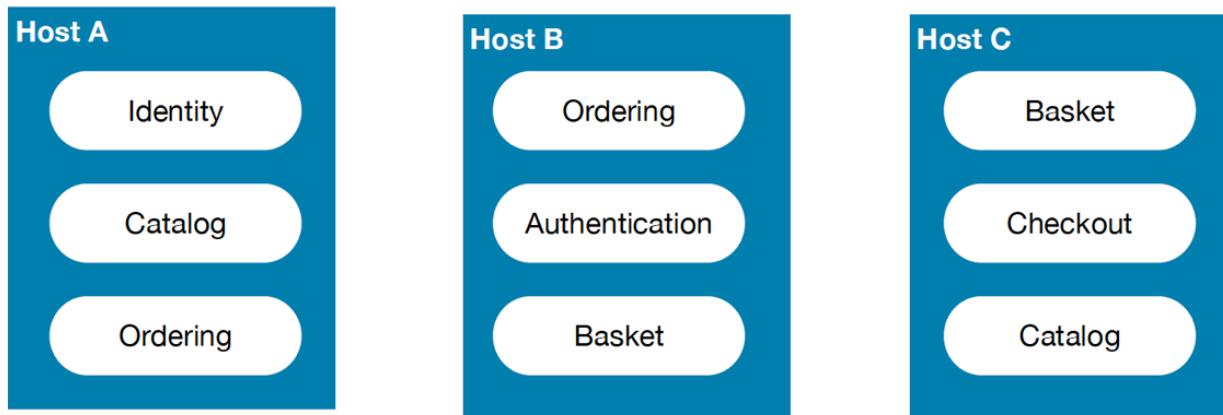
- ✓ **Utilisation efficace** des ressources.
- ✗ **Isolement limité** des instances de service.

PLUSIEURS INSTANCES PAR HÔTE



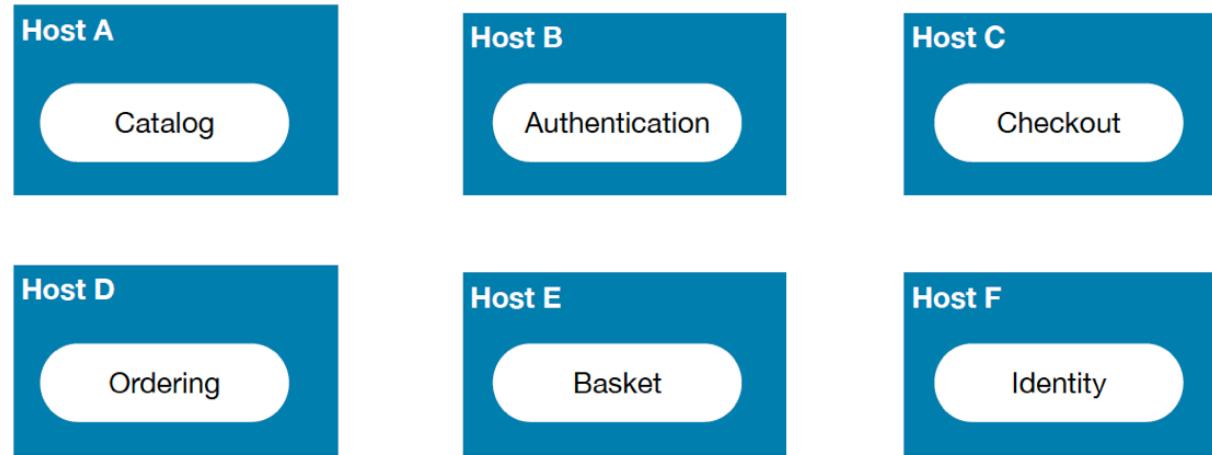
- ✓ **Utilisation efficace** des ressources.
- ✗ **Isolement limité** des instances de service.
- ✗ Difficile de **limiter les ressources** qu'un service utilise.

PLUSIEURS INSTANCES PAR HÔTE

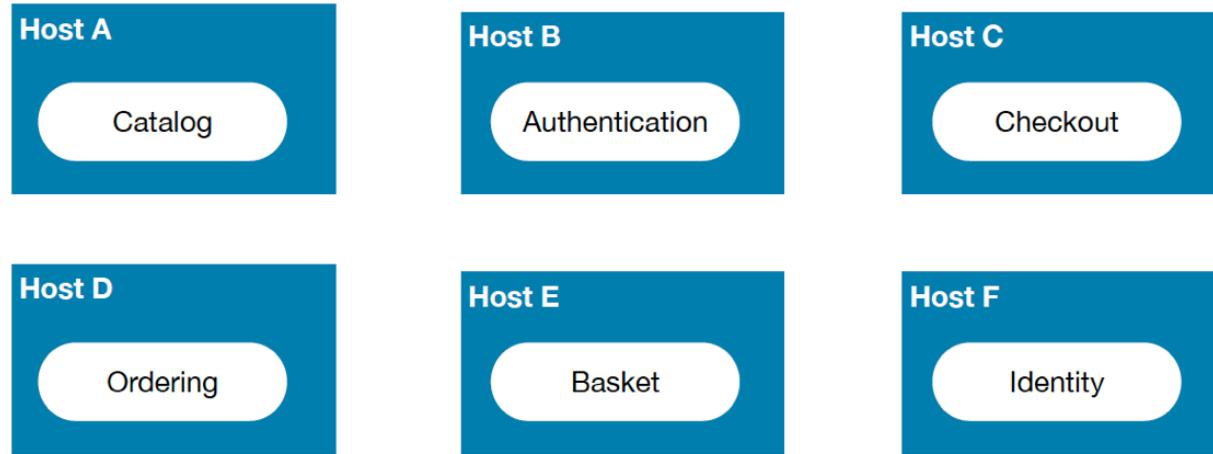


- ✓ **Utilisation efficace** des ressources.
- ✗ **Isolement limité** des instances de service.
- ✗ Difficile de **limiter les ressources** qu'un service utilise.
- ✗ **Déploiement difficile** :
les services sont écrits dans **différents langages** et **frameworks**.

INSTANCE DE SERVICE PAR HÔTE

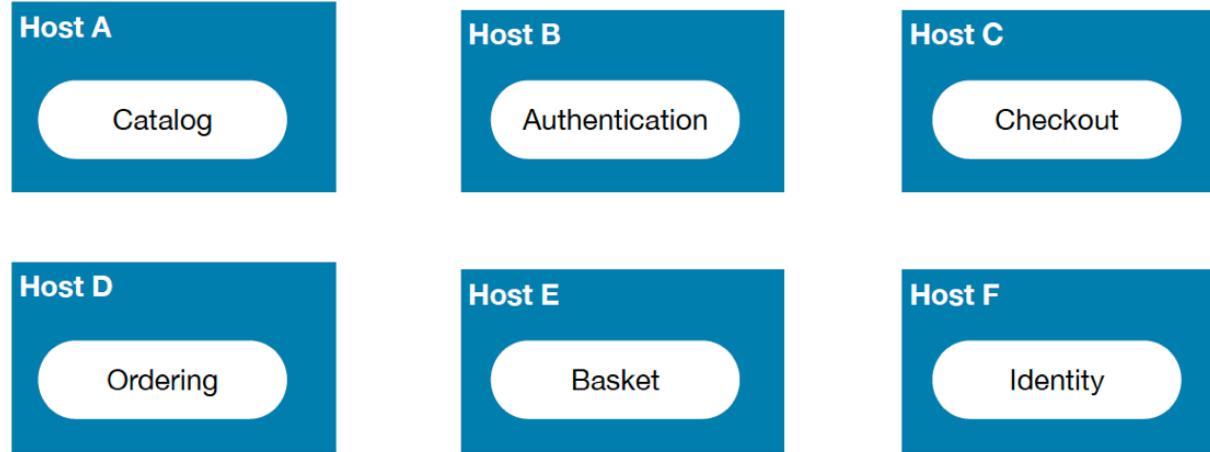


INSTANCE DE SERVICE PAR HÔTE



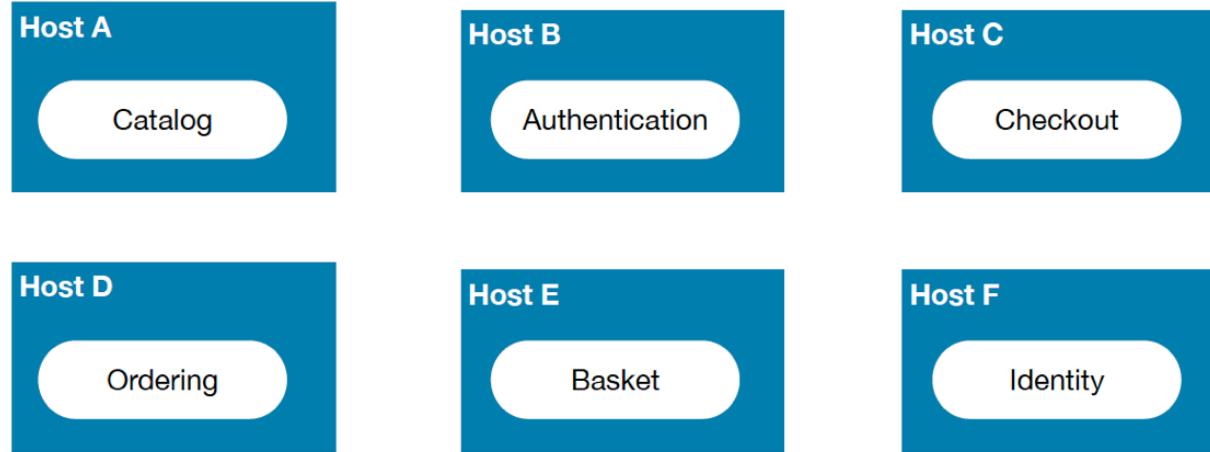
✓ **Isolement** entre les instances de service.

INSTANCE DE SERVICE PAR HÔTE



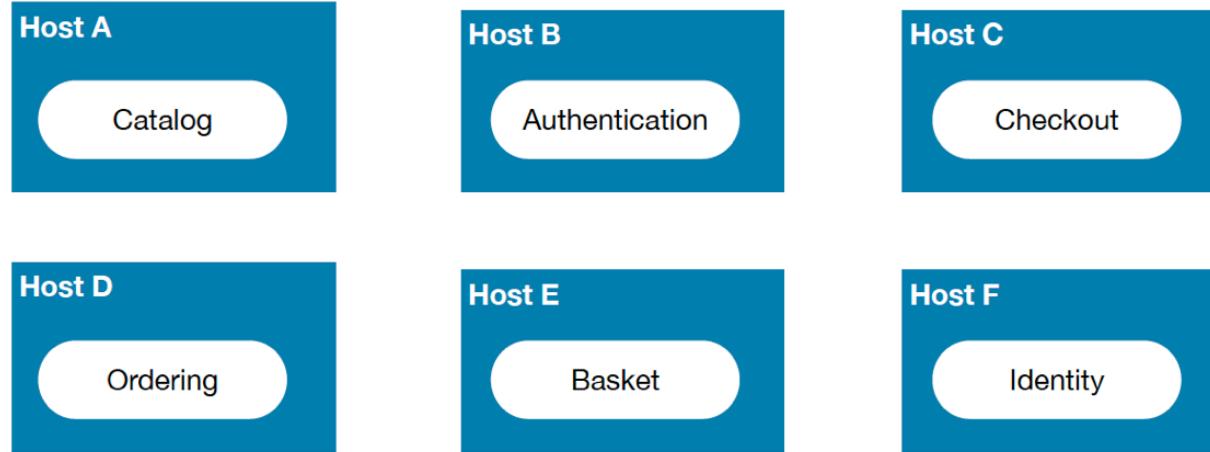
- ✓ **Isolement** entre les instances de service.
- ✓ **Monitoring facile** des ressources utilisées par un service.

INSTANCE DE SERVICE PAR HÔTE



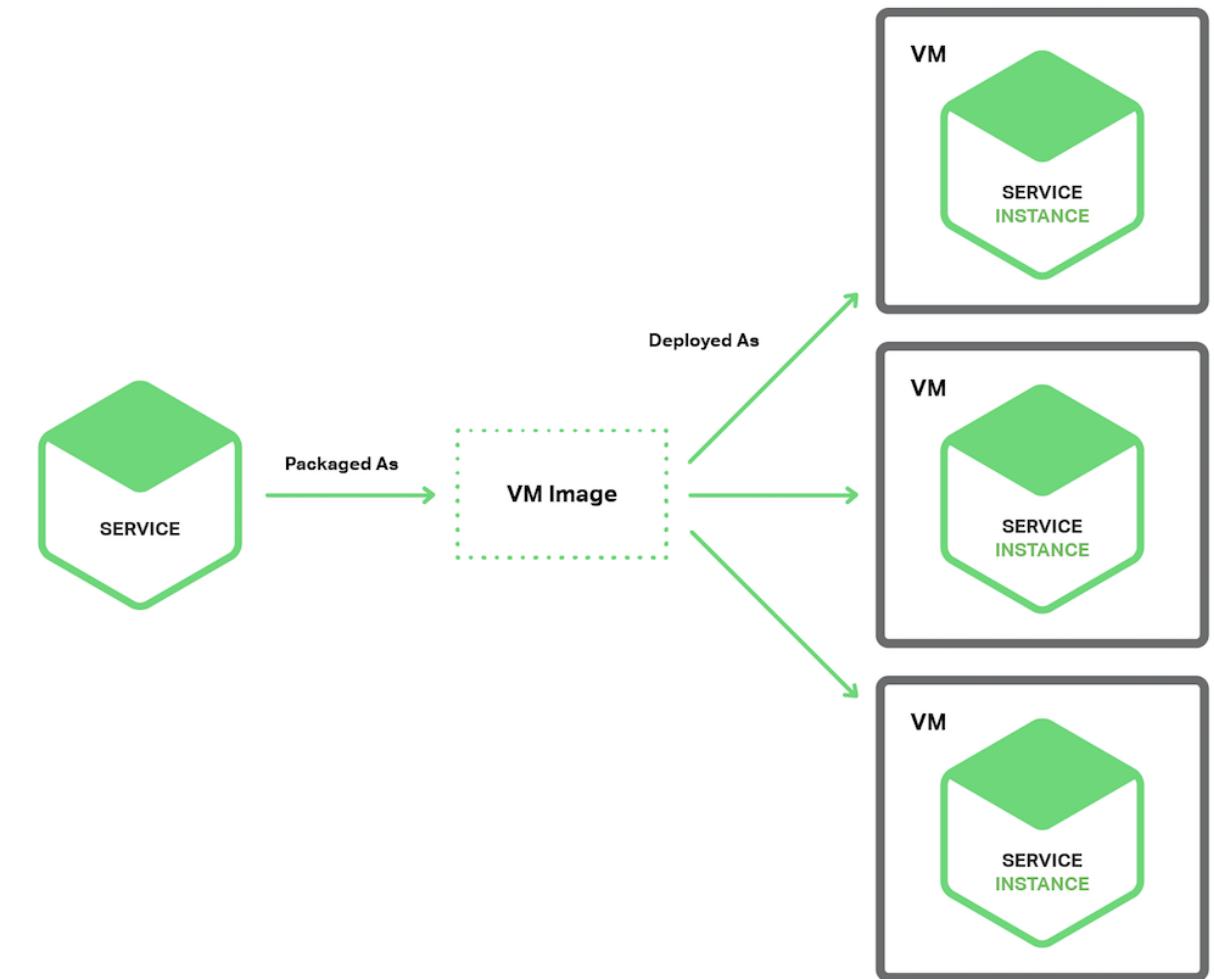
- ✓ **Isolement** entre les instances de service.
- ✓ **Monitoring facile** des ressources utilisées par un service.
- ✓ **Déploiement facile** :
pas d'exigences contradictoires entre deux prestations de service.

INSTANCE DE SERVICE PAR HÔTE



- ✓ **Isolement** entre les instances de service.
- ✓ **Monitoring facile** des ressources utilisées par un service.
- ✓ **Déploiement facile** :
pas d'exigences contradictoires entre deux prestations de service.
- ✗ **Utilisation inefficace** des ressources :
toutes les ressources d'un hébergeur sont dédiées à un seul service.

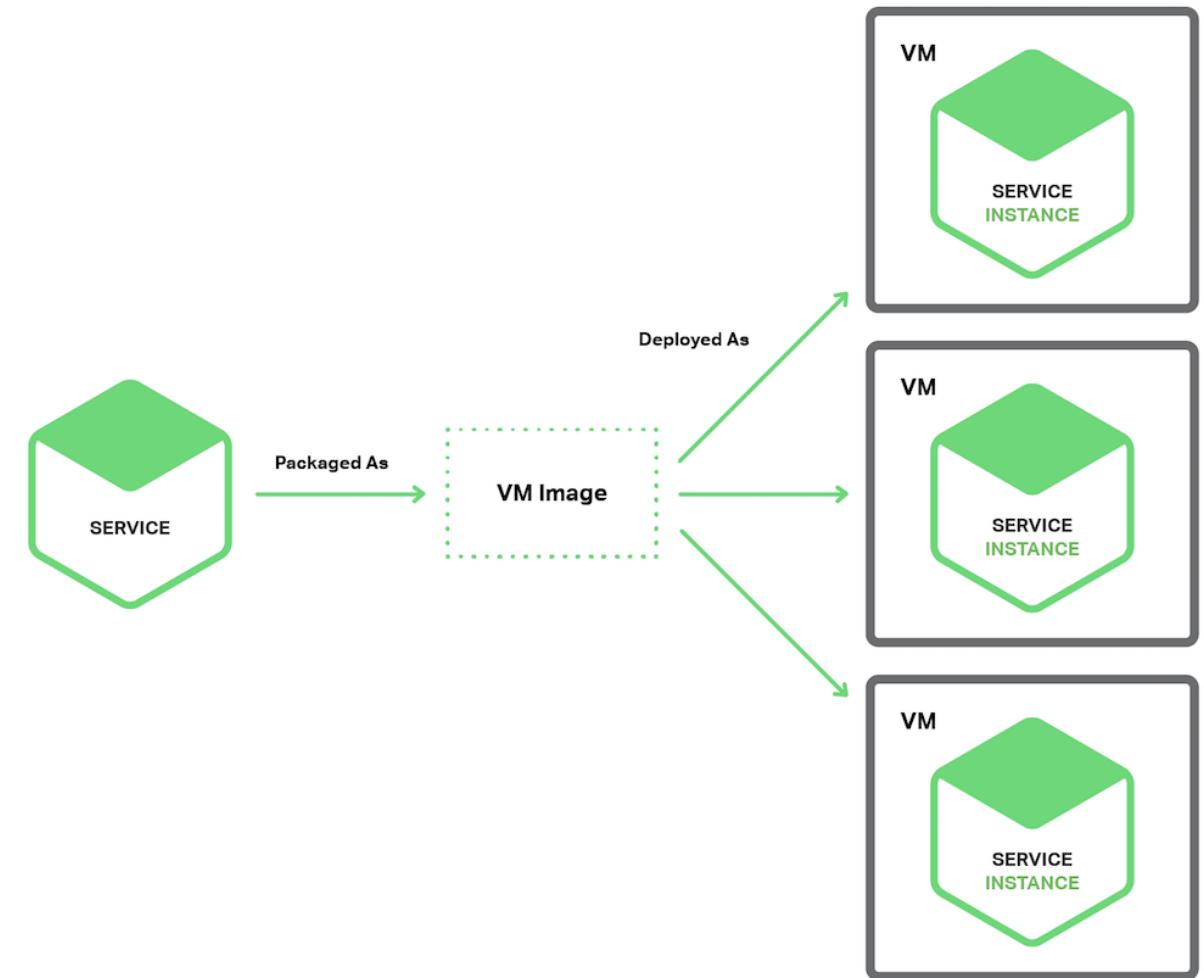
INSTANCE DE SERVICE PAR VM



Source : <https://www.nginx.com/>

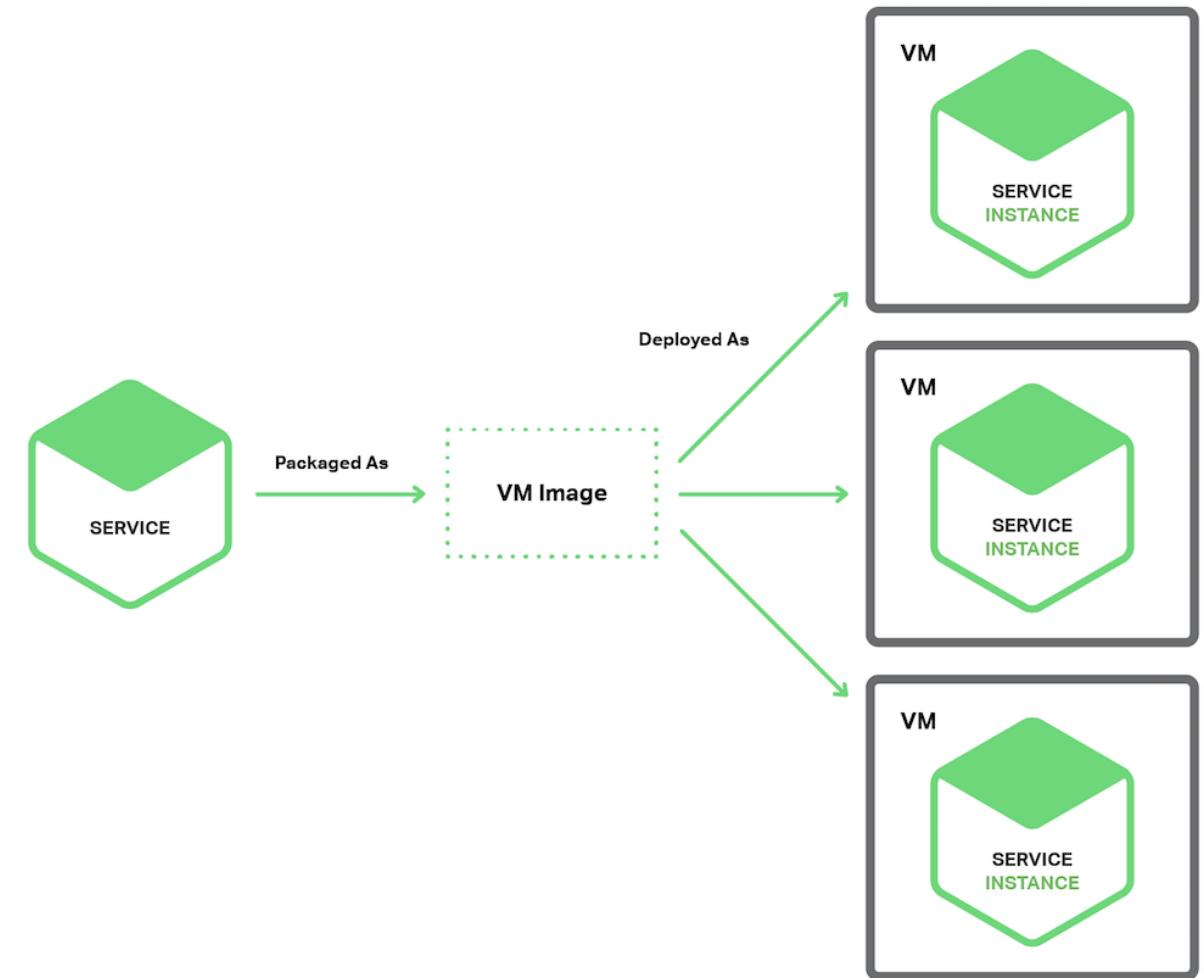
INSTANCE DE SERVICE PAR VM

- Chaque microservice est installé en tant que VM.



INSTANCE DE SERVICE PAR VM

- Chaque microservice est installé en tant que VM.
- Solution adoptée par Netflix.



INSTANCE DE SERVICE PAR VM

INSTANCE DE SERVICE PAR VM

- ✓ **Isolement** complet.

INSTANCE DE SERVICE PAR VM

- ✓ **Isolement** complet.
- ✓ **Quantité fixe de ressources par VM** (CPU, mémoire, ...).

INSTANCE DE SERVICE PAR VM

- ✓ **Isolement** complet.
- ✓ **Quantité fixe de ressources par VM** (CPU, mémoire, ...).
- ✓ **Les détails technologiques** sont **cachés dans la VM**.
 - 👉 tous les services sont démarrés et arrêtés de la même manière.

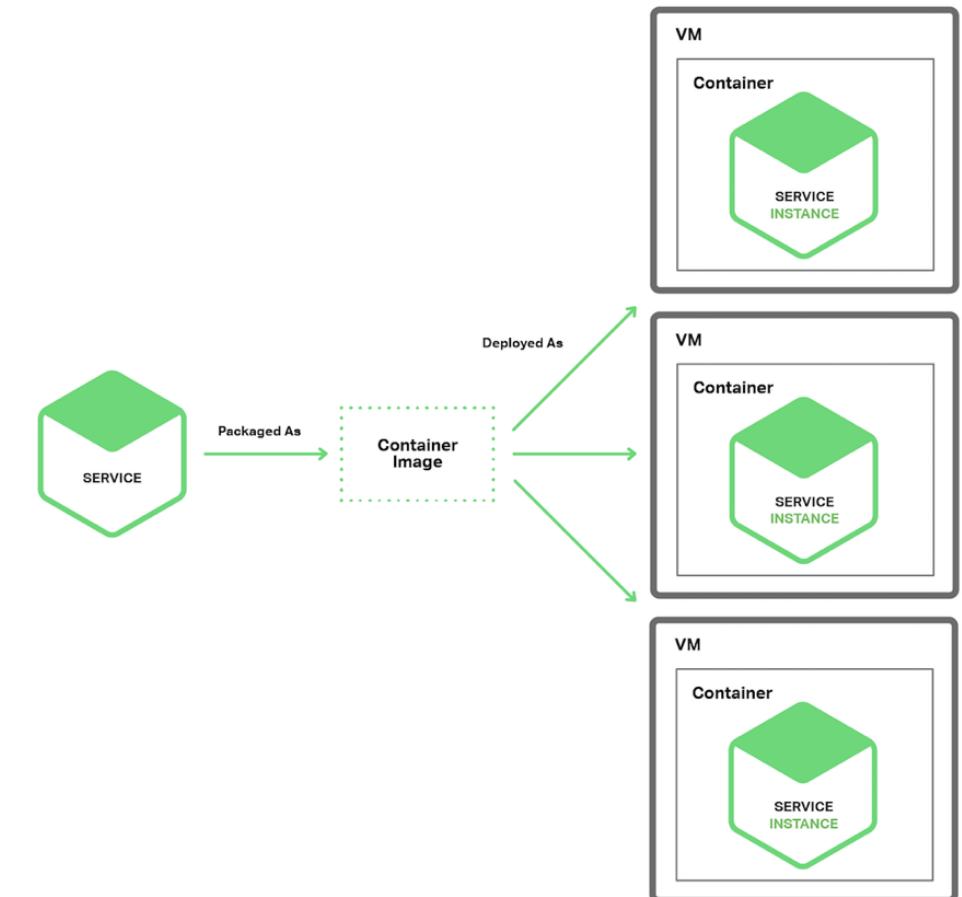
INSTANCE DE SERVICE PAR VM

- ✓ **Isolement** complet.
- ✓ **Quantité fixe de ressources par VM** (CPU, mémoire, ...).
- ✓ **Les détails technologiques** sont **cachés dans la VM**.
 - 👉 tous les services sont démarrés et arrêtés de la même manière.
- ✓ Infrastructure **Cloud mature**
 - 👉 équilibrage de charge et autoscaling.

INSTANCE DE SERVICE PAR VM

- ✓ **Isolement** complet.
- ✓ **Quantité fixe de ressources par VM** (CPU, mémoire, ...).
- ✓ **Les détails technologiques** sont **cachés dans la VM**.
 - 👉 tous les services sont démarrés et arrêtés de la même manière.
- ✓ Infrastructure **Cloud mature**
 - 👉 équilibrage de charge et autoscaling.
- ✗ Les **images de VM sont lentes** à créer et à instancier.
 - 👉 Une machine virtuelle contient généralement un OS complet
 - 👉 Des solutions existent pour générer des images légères

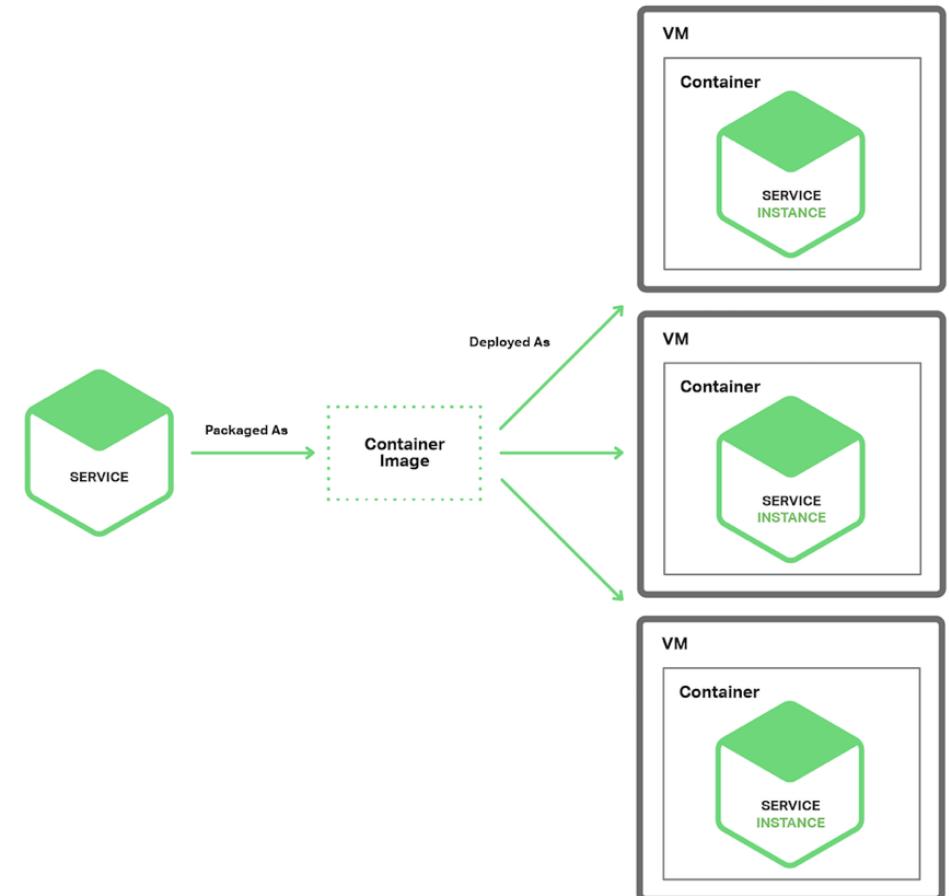
INSTANCE DE SERVICE PAR CONTENEUR



Source : <https://www.nginx.com/>

INSTANCE DE SERVICE PAR CONTENEUR

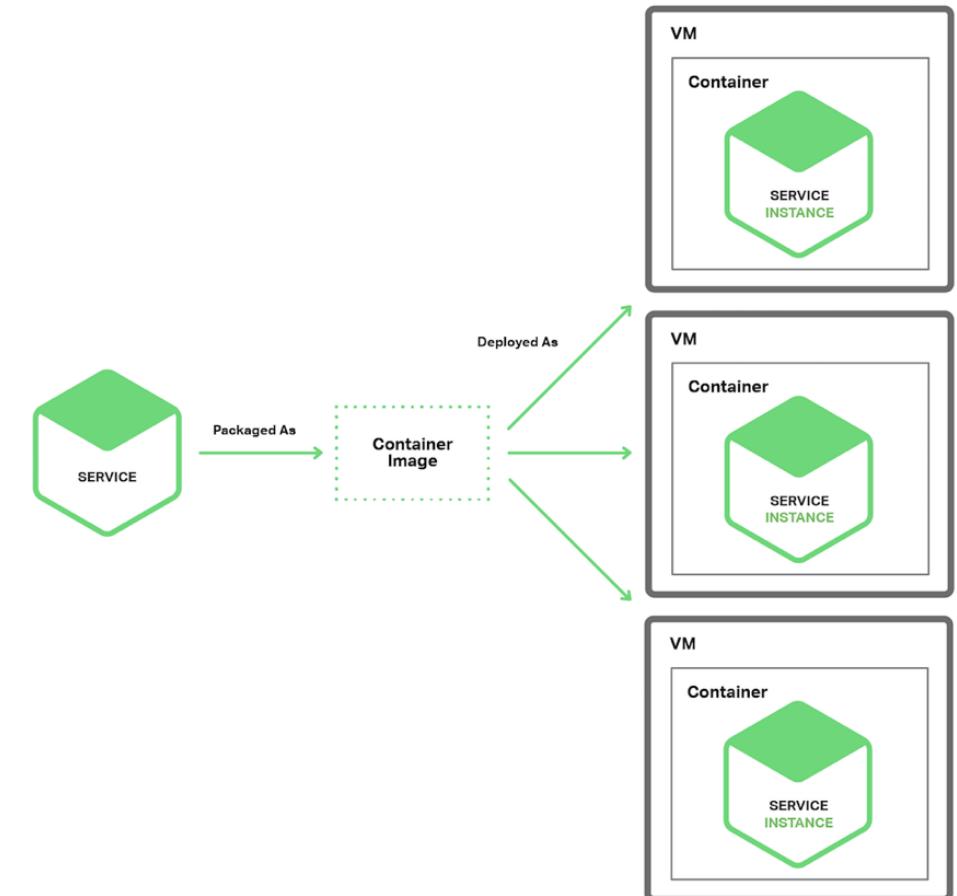
- **Microservice** installé sous forme d'image de **conteneur** (**VM légère**).
👉 ils contiennent ce qu'il faut pour exécuter le service



Source : <https://www.nginx.com/>

INSTANCE DE SERVICE PAR CONTENEUR

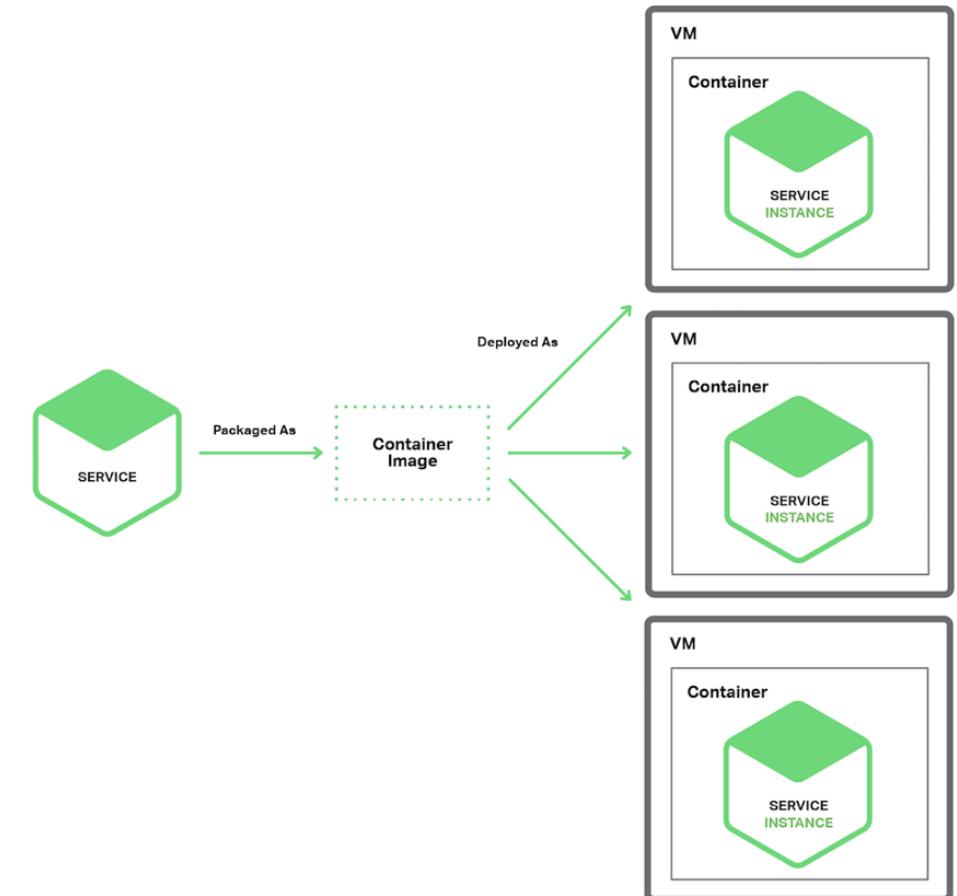
- **Microservice** installé sous forme d'image de **conteneur** (**VM légère**).
👉 ils contiennent ce qu'il faut pour exécuter le service
- **Les conteneurs** peuvent s'exécuter sur une **VM** ou un **serveur physique**.



Source : <https://www.nginx.com/>

INSTANCE DE SERVICE PAR CONTENEUR

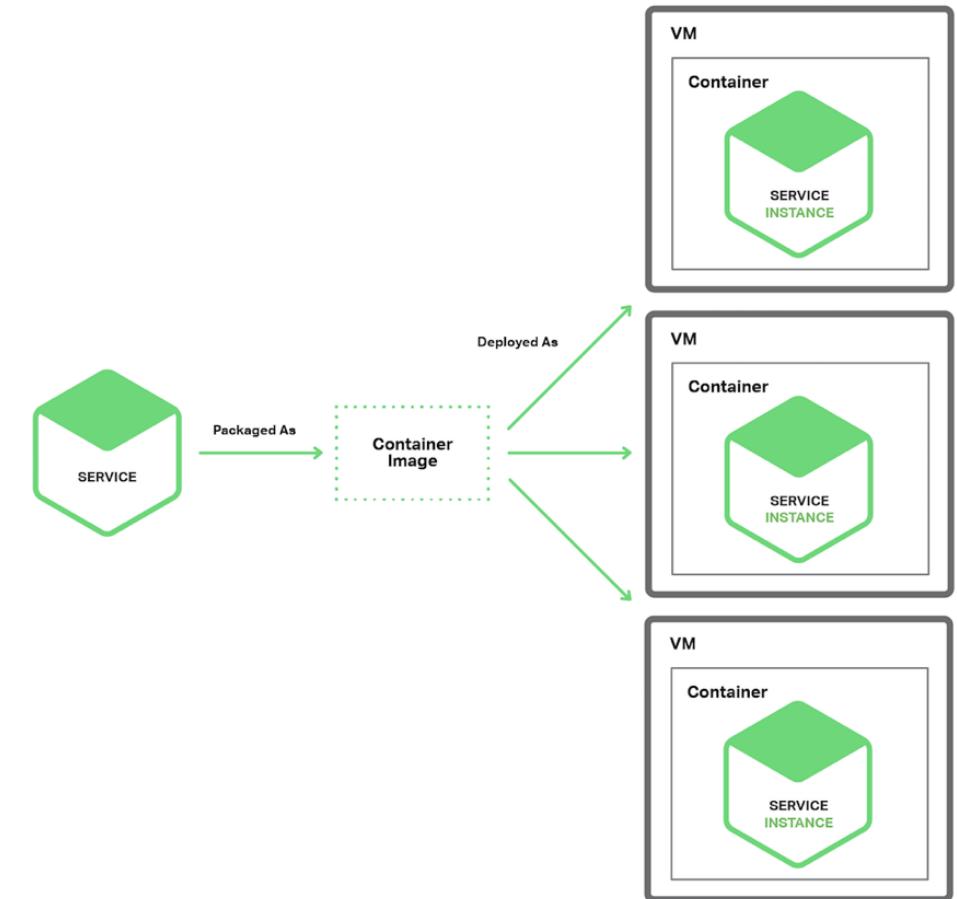
- **Microservice** installé sous forme d'image de **conteneur (VM légère)**.
👉 ils contiennent ce qu'il faut pour exécuter le service
- **Les conteneurs** peuvent s'exécuter sur une **VM** ou un **serveur physique**.
- Tous les avantages des VM et ...



Source : <https://www.nginx.com/>

INSTANCE DE SERVICE PAR CONTENEUR

- **Microservice** installé sous forme d'image de **conteneur (VM légère)**.
👉 ils contiennent ce qu'il faut pour exécuter le service
- **Les conteneurs** peuvent s'exécuter sur une **VM** ou un **serveur physique**.
- Tous les avantages des VM et ... sont **rapides à construire**



Source : <https://www.nginx.com/>

SERVERLESS DEPLOYMENT

SERVERLESS DEPLOYMENT

- La **serverless computing platform** (**Google, Microsoft Azure, ...**) exécute autant d'instances du service que nécessaire.

SERVERLESS DEPLOYMENT

- La **serverless computing platform** (**Google, Microsoft Azure, ...**) exécute autant d'instances du service que nécessaire.
- **Aucune visibilité** sur l'infrastructure sous-jacente.

SERVERLESS DEPLOYMENT

- La **serverless computing platform** (**Google, Microsoft Azure, ...**) exécute autant d'instances du service que nécessaire.
 - **Aucune visibilité** sur l'infrastructure sous-jacente.
- ✓ pas besoin de configurer l'infrastructure sous-jacente.

SERVERLESS DEPLOYMENT

- La **serverless computing platform** (**Google, Microsoft Azure, ...**) exécute autant d'instances du service que nécessaire.
- **Aucune visibilité** sur l'infrastructure sous-jacente.
 - ✓ pas besoin de configurer l'infrastructure sous-jacente.
 - ✓ payez à la demande, au lieu de payer pour l'infrastructure.

SERVERLESS DEPLOYMENT

- La **serverless computing platform** (**Google, Microsoft Azure, ...**) exécute autant d'instances du service que nécessaire.
- **Aucune visibilité** sur l'infrastructure sous-jacente.
 - ✓ pas besoin de configurer l'infrastructure sous-jacente.
 - ✓ payez à la demande, au lieu de payer pour l'infrastructure.
- ✗ inapte aux services de longue durée.

SERVERLESS DEPLOYMENT

- La **serverless computing platform** (**Google, Microsoft Azure, ...**) exécute autant d'instances du service que nécessaire.
- **Aucune visibilité** sur l'infrastructure sous-jacente.
 - ✓ pas besoin de configurer l'infrastructure sous-jacente.
 - ✓ payez à la demande, au lieu de payer pour l'infrastructure.
 - ✗ inapte aux services de longue durée.
 - ✗ aucun contrôle sur les performances.

ON-DEMAND COMPUTING

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**
 - **mutualisation** des ressources entre départements

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**
 - **mutualisation** des ressources entre départements
 - la **DSI** devient un prestataire de service

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**
 - mutualisation des ressources entre départements
 - la **DSI** devient un prestataire de service
- **Cloud public (Amazon, MS Azure, ...)**

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**
 - mutualisation des ressources entre départements
 - la **DSI** devient un prestataire de service
- **Cloud public (Amazon, MS Azure, ...)**
 - nouveau modèle économique: *Pay As You Go*

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**
 - mutualisation des ressources entre départements
 - la **DSI** devient un prestataire de service
- **Cloud public (Amazon, MS Azure, ...)**
 - nouveau modèle économique: *Pay As You Go*
- **Cloud hybride**

ON-DEMAND COMPUTING

La **virtualisation** à la base du **Cloud Computing**

- **Cloud privé**
 - mutualisation des ressources entre départements
 - la **DSI** devient un prestataire de service
- **Cloud public (Amazon, MS Azure, ...)**
 - nouveau modèle économique: *Pay As You Go*
- **Cloud hybride**
 - en cas de dépassement de capacité des ressources propres
 - 👉 extension au **cloud public**

FIN

- Retour à l'accueil
- Retour au plan