Principes des systèmes d'exploitation et programmation concurrente

Introduction

UFR IM^{2}AG

M1 MEEF NSI 2022-2023

Renaud Lachaize

https://m1-meef-nsi-info701-os.gricad-pages.univ-grenoble-alpes.fr

Crédits et remerciements

- **Le contenu de ce support de cours est partiellement inspiré, voire emprunté aux travaux d'autres personnes :**

  - Vincent Danjean, Sacha Krakowiak, Arnaud Legrand, Vania Marangozova-Martin (UFR IM^{2}AG)
  - David Mazières (Stanford University)
  - Randall Bryant, David O'Hallaron, Gregory Kesden, Markus Püschel (Carnegie Mellon University)
  - Gernot Heiser (UNSW)

  - Ouvrages de référence (voir détails sur la page web) :
    - Silberschatz et al. Operating systems concepts with Java.
    - Tanenbaum. Modern operating systems.
    - Bryant and O'Hallaron. Computer systems: a programmer's perspective.

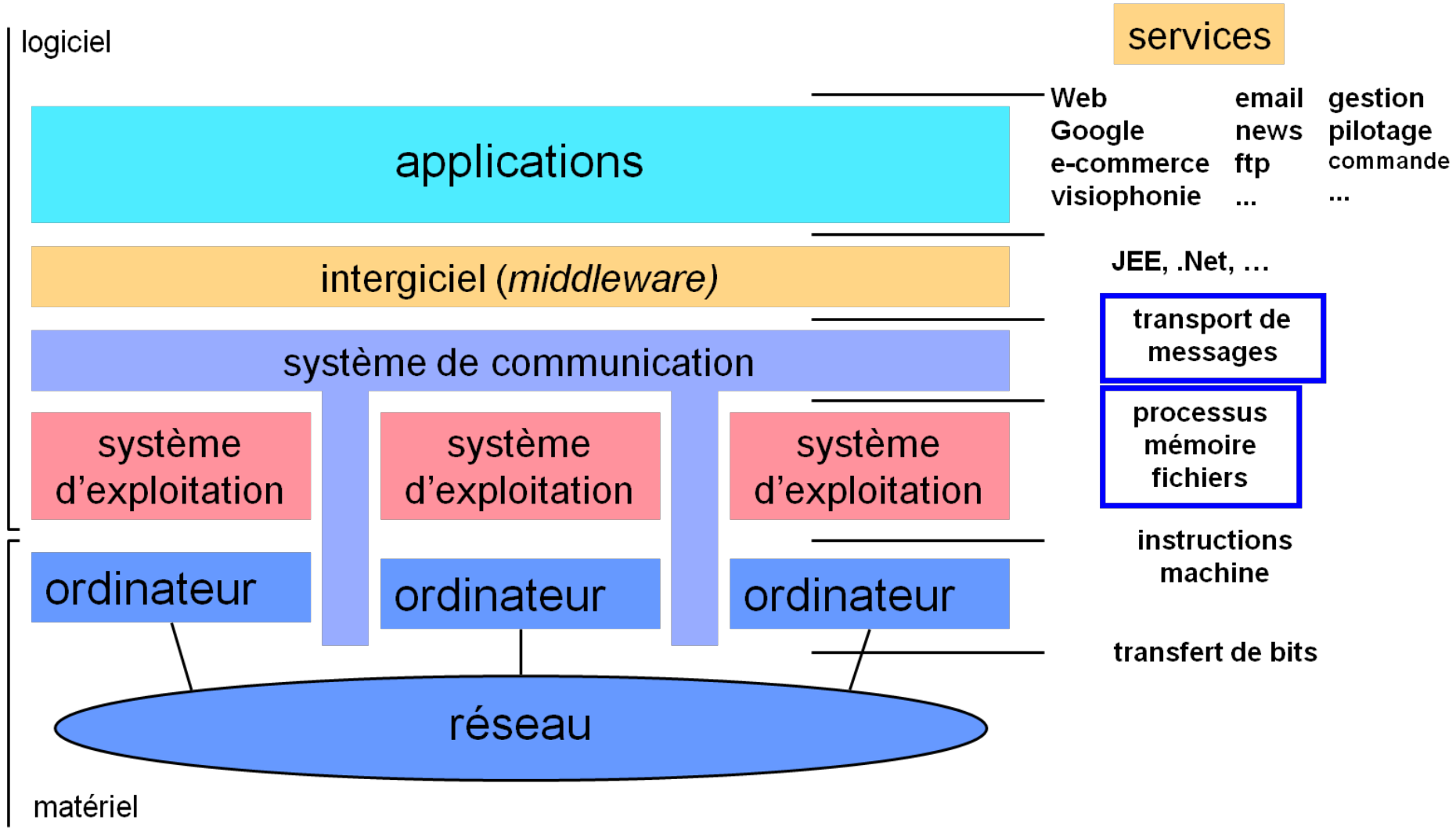
Objectifs du cours

- **Comprendre les principaux concepts et mécanismes des systèmes d'exploitation, ainsi que leurs éléments de réalisation**
- **Mieux comprendre les interactions entre les couches matérielles et logicielles**
- **Mieux comprendre les interactions entre couches logicielles**
  - exemples : environnements Java, Python
- **Maîtriser la programmation concurrente**
- **En Résumé, acquérir des connaissances nécessaires pour :**
  - Programmer (de manière fiable et efficace) et structurer des logiciels complexes
  - Appréhender des domaines connexes (intergiciels, systèmes répartis ...)

Plan

- Introduction : position et rôle d'un système d'exploitation
- Notions de base: processus, protection et partage de ressources
- Threads
- Machines virtuelles

Composants d'un système informatique



Qu’est-ce qu’un système d’exploitation ?

- **Au sens large : les couches logicielles intermédiaires entre le matériel (processeurs et périphériques) et les applications**

- **Plus précisément : les couches logicielles de plus bas niveau sur une machine**
  - Noyau
  - Bibliothèques de base et quelques programmes utilitaires

- **Deux rôles principaux et complémentaires**
  - Adaptation d’interface
  - Gestion de ressources

- **Présent sur la plupart des équipements munis d’un processeur**
  - Serveurs, ordinateurs personnels, tablettes, téléphones, carte à puces ...

Adaptation d'interface

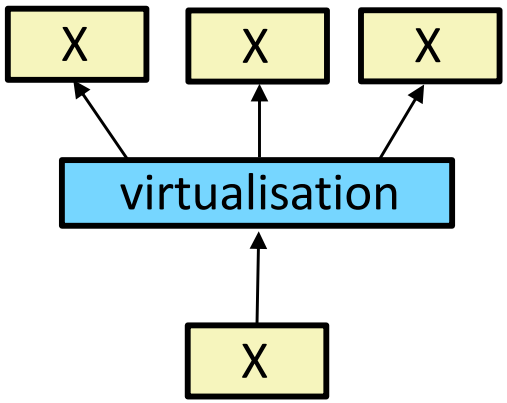
- **But : simplifier l'écriture et l'exécution (lancement, débogage, administration ...) de programmes sur une machine**

- **Pour cela, le système d'exploitation, décharge les programmeurs/utilisateurs d'applications de la gestion de certains aspects complexes :**
  - Les interactions avec le matériel (interfaces de très bas niveau, diversité et hétérogénéité des périphériques)
  - Les limites physiques des machines et leurs variations d'une machine à l'autre (taille de la mémoire vive, nombre de processeurs/cœurs)
  - Le partage des ressources entre applications/utilisateurs (cf. gestion de ressources)

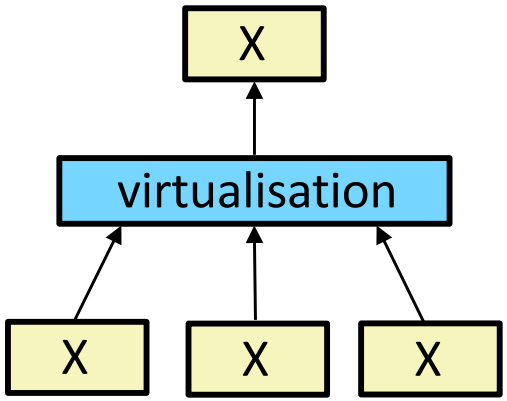
- **Idée générale : masquer les détails complexes derrière des abstractions, souvent de plus haut niveau**

Virtualisation

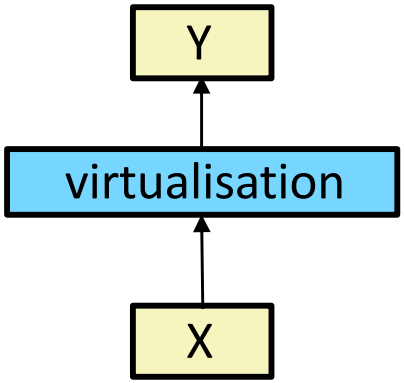
- **Idée générale : abstraire les ressources sous-jacentes**
- **Il existe différentes formes de virtualisation**
  - (qui peuvent être combinées)



(a) Multiplexage



(b) Agrégation



(c) Émulation

Inspired (with modifications) from books and slides by: J. Salzer & M.F. Kaashoek, E. Bugnion & J. Nieh & D. Tsafrir, S. Krakowiak

Gestion de ressources

- **But: permettre un fonctionnement stable/sécurisé/efficace/équitable (...) des applications**
  - malgré leur exécution concurrente et des demandes/besoins potentiellement contradictoires
  - malgré d'éventuels bogues/pannes/tentatives de malveillance

- **Plusieurs types de ressources**
  - Physiques : processeur(s), mémoire, périphériques, énergie ...
  - Logiques : programmes, données, communications ...
- **Plusieurs aspects**
  - Allocation
  - Protection
  - Partage (spatial et temporel)

- **Idée générale : la gestion de chaque ressource/aspect repose sur des mécanismes (génériques) et sur des politiques (configurables)**

Fonctions d'un système d'exploitation (1/2)

- **Gestion d'activités**
  - Déroulement de l'exécution
  - Evénements particuliers (matériel, erreurs, interactions entre activités ...)
  - Abstraction clé : processeur => processus

- **Gestion d'information**
  - Accès dynamique (exécution)
  - Partage
  - Conservation permanente
  - Abstractions clés :
    - Mémoire principale => mémoire virtuelle
    - Disque => fichiers

Fonctions d'un système d'exploitation (2/2)

- **Gestion des communications**
  - Interface avec l'utilisateur
  - Interface avec les périphériques d'entrées/sorties
  - Interactions avec d'autres machines via le réseau
  - Abstraction clé : périphérique => flot d'entrées/sorties

- **Protection**
  - Pour les ressources matérielles et les informations
  - Notions clés : domaines de protection et politiques associées

Interfaces d'un système d'exploitation (1/2)

- **Un système exporte typiquement deux types d'interfaces**
  - Une interface de commande
  - Une interface programmatique

- **Interface de commande**
  - Conçue pour les interactions avec les utilisateurs
  - Diverses formes : interfaces textuelles (shells), interfaces graphiques
  - Composée d'un ensemble de commandes
    - Exemple : supprimer un fichier
    - Version textuelle (shell Unix) : **`rm myfile.txt`**
    - Version graphique : faire glisser l'icône du fichier vers la corbeille

Interfaces d'un système d'exploitation (2/2)

- **Interface programmatique**
  - Utilisée/invoquée par les applications qui s'exécutent sur le système
    - Y compris les programmes qui implémentent les interfaces de commande
  - Composée d'un ensemble de procédures/fonctions
    - Bibliothèques
    - Appels système (voir détails un peu plus loin)
  - Définie à deux niveaux
    - Au niveau du code source : *Application Programming Interface* (API) – Exemple : norme POSIX
    - Au niveau du code binaire : *Application Binary Interface* (ABI) – Exemple : ABI Linux x86 32bits

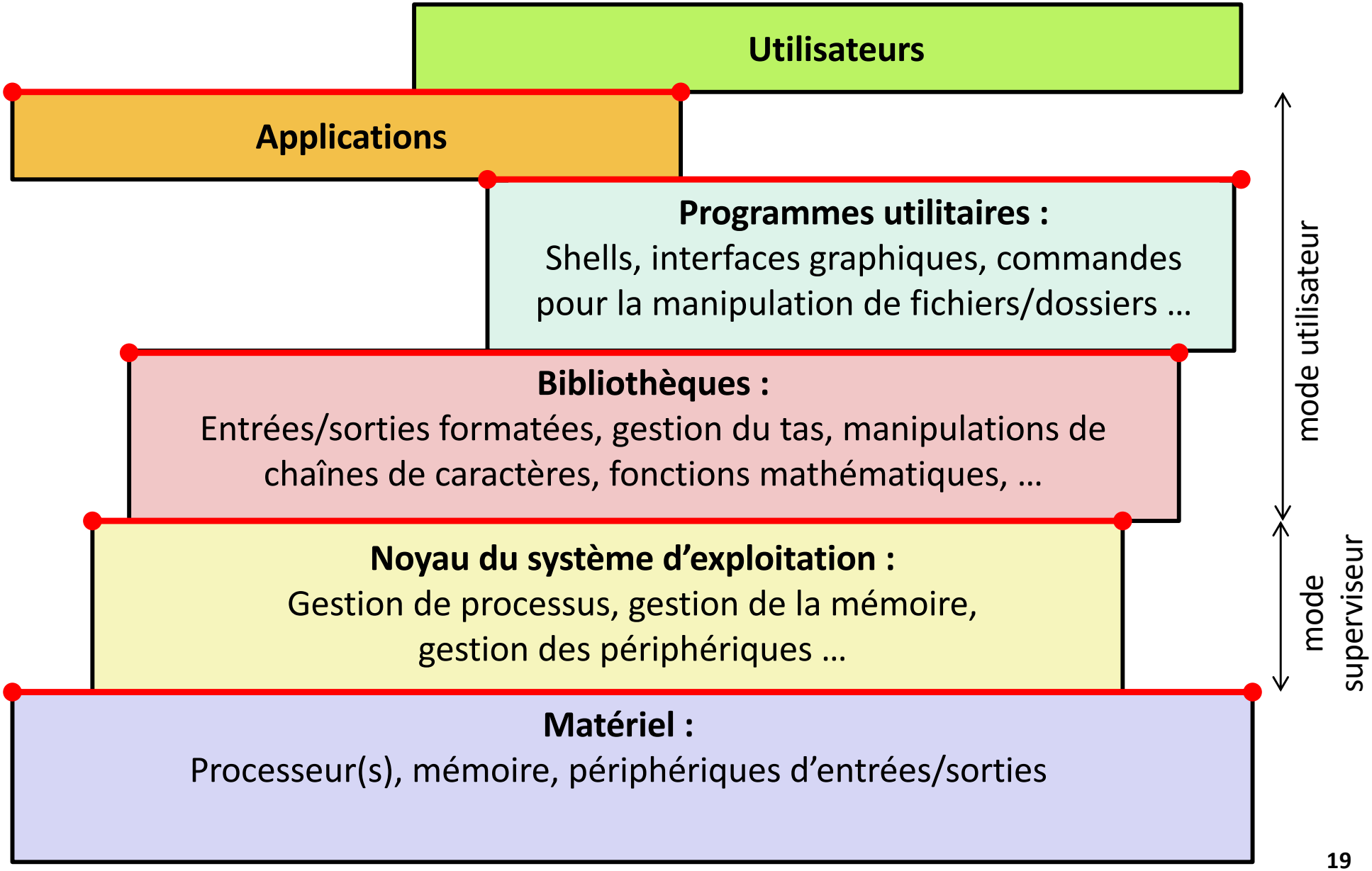
Un peu d'histoire

- **Premiers systèmes d'exploitation**
  - De simples bibliothèques de code permettant de simplifier l'écriture des programmes
  - Par exemple, pour la gestion des périphériques

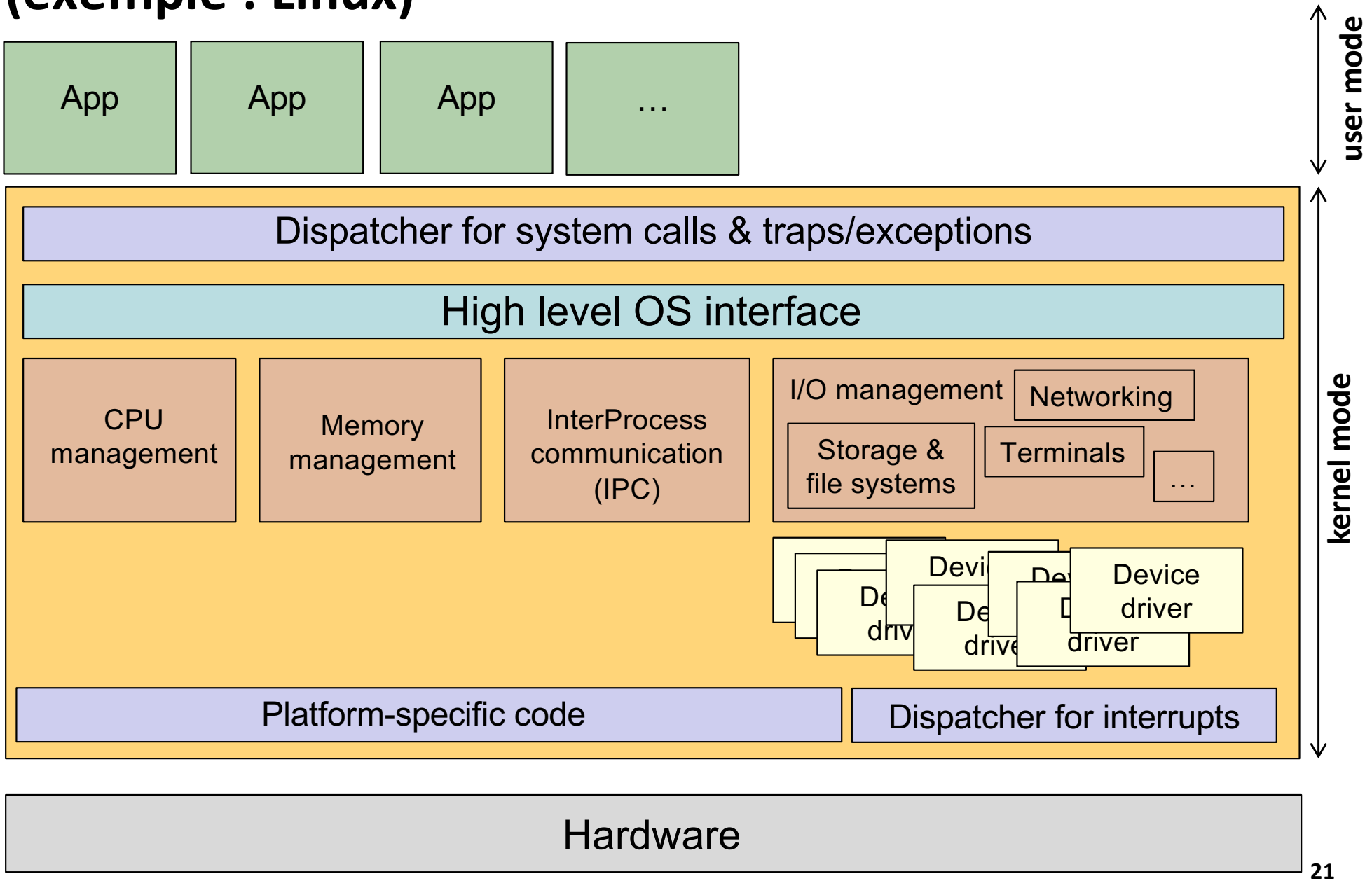
- **2ème étape : prise en compte du rôle central du système d'exploitation**
  - Un utilisateur/une application ne doit pas rendre le système global inutilisable
  - Introduction d'une nouvelle interface (matérielle et logicielle) pour protéger le code et les données critiques du système

- **3ème étape : amélioration de l'utilisation des ressources et de la réactivité**
  - Ne pas nécessairement attendre la terminaison d'une application avant d'en lancer une autre
  - Empêcher les interférences entre différentes applications

Structure typique d'un système d'exploitation



Structure typique d'un noyau monolithique (exemple : Linux)



Structure de type micro-noyau (exemple : L4)