

M1 SAR 2017-2018
Programmation Noyau Linux
Cours 1 - Introduction

Kevin Mambu

February 1, 2018

Contents

1	Introduction	3
1.1	Avertissement	3
1.2	Présentation	3
2	Systèmes d'Exploitation - Rappels	3
2.1	Noyaux monolithiques	4
2.2	Micro-noyaux	5
2.3	Noyaux modulaires monolithiques	6
2.4	Micro-noyaux modulaires	7
2.5	Exo-noyau	8
2.6	Uni-Kernel	9
3	Histoire de Linux	10
3.1	L'origine de "Unix"	10

1 Introduction

1.1 Avertissement

Il s'agit d'une UE très exigeante sur la maîtrise d'un noyau et d'oeuvres avancées du C.

1.2 Présentation

L'objectif de l'UE est de maîtriser à terme le développement au sein d'un Noyau Linux, le déboguage avec des outils adaptés et la programmation par patch et modules.

- Examen final : 40%
- Projet & Exercices : 60%

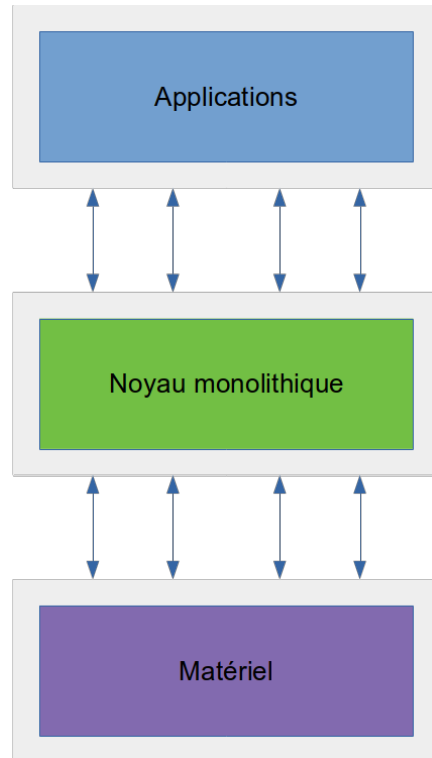
Deux choses à noter :

- Ce cours serait un très bon cours de génie logiciel : plusieurs milliers de contributeurs, un code très stable et viable car codé de manière très codifiée.
- Même sans développer dans le noyau Linux plus tard : il est important de connaître et de comprendre le système d'exploitation et les outils & bonnes pratiques.

2 Systèmes d'Exploitation - Rappels

Qu'est ce qu'un système d'exploitation, en essence? \Rightarrow Un gestionnaire d'interruption.
Il y a quatre types de Noyau:

2.1 Noyaux monolithiques



Un noyau compilé statiquement, sur un seul binaire (UNICS V6 par exemple).
WARN : un noyau monolithique ne veut pas dire un code monolithique. Le noyau fournit aux applications utilisateurs une abstraction concernant les abstractions du matériel. Communiquer de l'application au matériel se passe par le biais d'interruptions \Rightarrow mapper différentes interruptions à différentes actions.

Un OS se distingue par deux modes d'exécution différents : le mode système et le mode utilisateur. Le premier possède des avantages tels que :

- Zones mémoire restreintes
- Instructions privilégiées
- Pile spécifique
- Accès à certains registres

Intérêts :

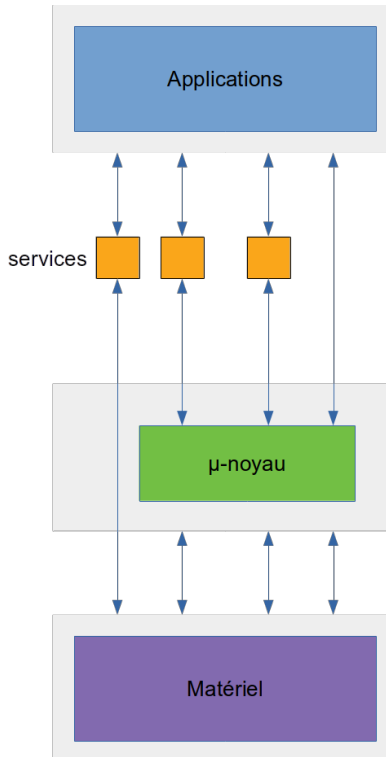
- Facile à optimiser car sur la même codebase \Rightarrow performant
- Facile à sécuriser

Inconvénients :

- Empreinte mémoire assez importante quand non-dédiée
- Beaucoup de fonctionnalités \Rightarrow Beaucoup de failles potentielles

Utilisation du noyau monolithique \Rightarrow embarqué, applications RT ou spécifique.

2.2 Micro-noyaux



Idée : Réduire le noyau à un set de fonctionnalités fondamentales, et mettre les autres services en mode utilisateurs. On réduit la surface d'attaque du noyau en dispatchant dans le mode utilisateur (exemple : filesystems)

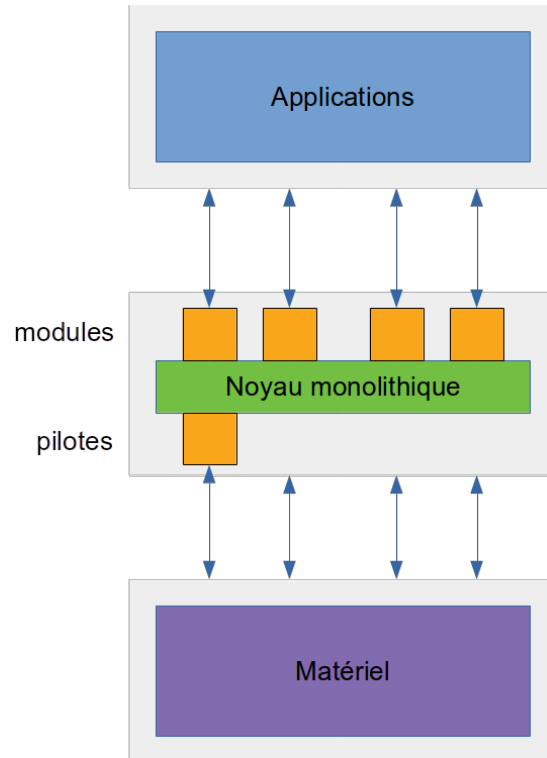
Intérêts :

- Empreinte mémoire très réduite
- Surface d'attaque très restreinte
- Fonctionnalités en mode utilisateur \Rightarrow execution en mode utilisateur

Inconvénients :

- Beaucoup d'appels systèmes \Rightarrow impact sur les performances
- Codebase dispatchée \Rightarrow plus difficile à tenir à jour

2.3 Noyaux modulaires monolithiques



Avoir un noyau monolithique auquel on peut charger des modules. Intérêts :

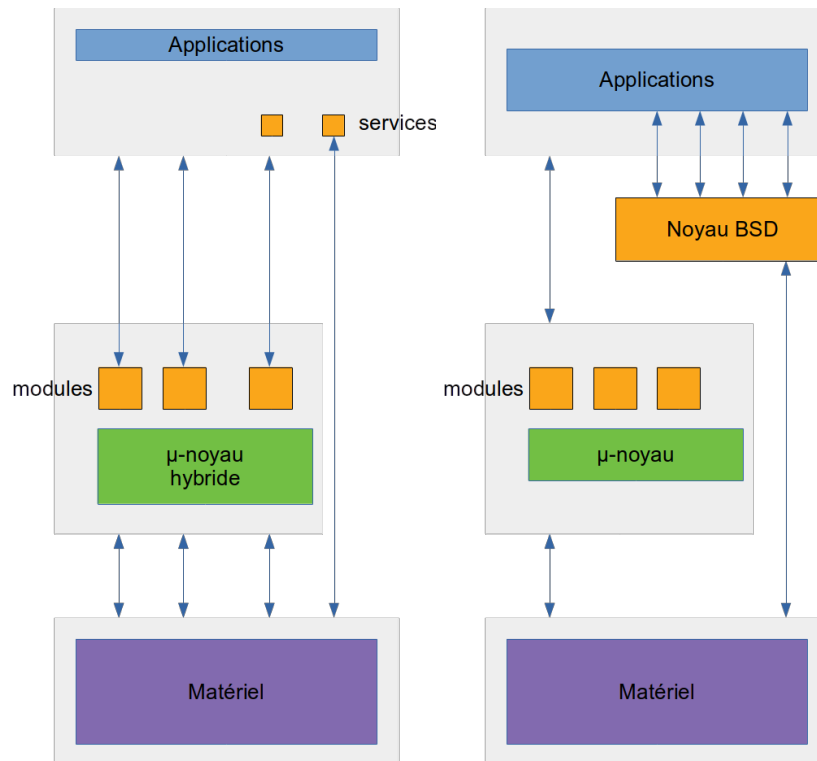
- Interaction directe avec le matériel optimisé
- Possibilité d'opti pour une architecture
- facile d'extension et de portabilité
- Modules directement en mode privilégié

Inconvénients :

- Pas très portable
- Modules tiers en mode privilégié \Rightarrow sécurité compromise!

Nb : il est possible de hard link des modules pour faire de la programmation monolithique.

2.4 Micro-noyaux modulaires

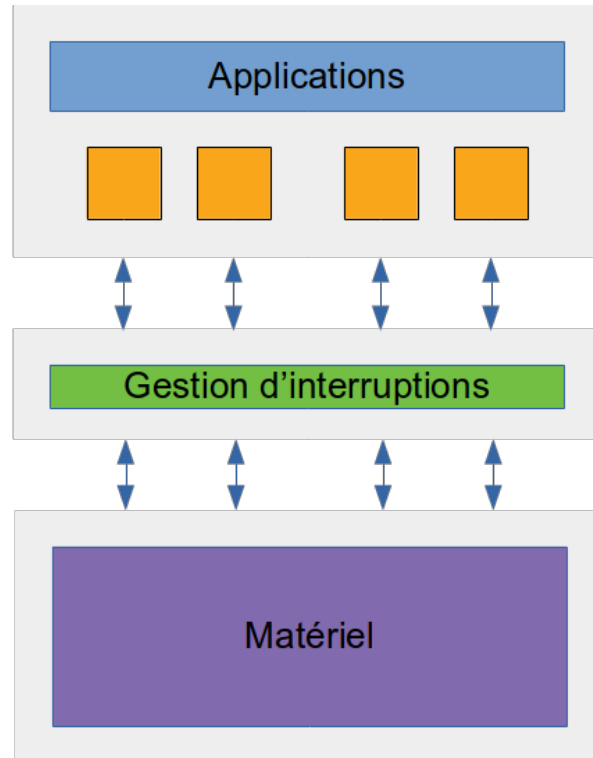


Possibilité d'avoir d'une part des services haut-niveau opérés en mode utilisateur et un micro-noyau modulaire. Cela permet de combiner les avantages de modularités des micro-noyaux et des noyaux monolithiques modulaires.

On pourrait dire que parce que glibc est l'API de choix des développeurs sous Linux, Linux est quelque part un micro-noyau modulaire.

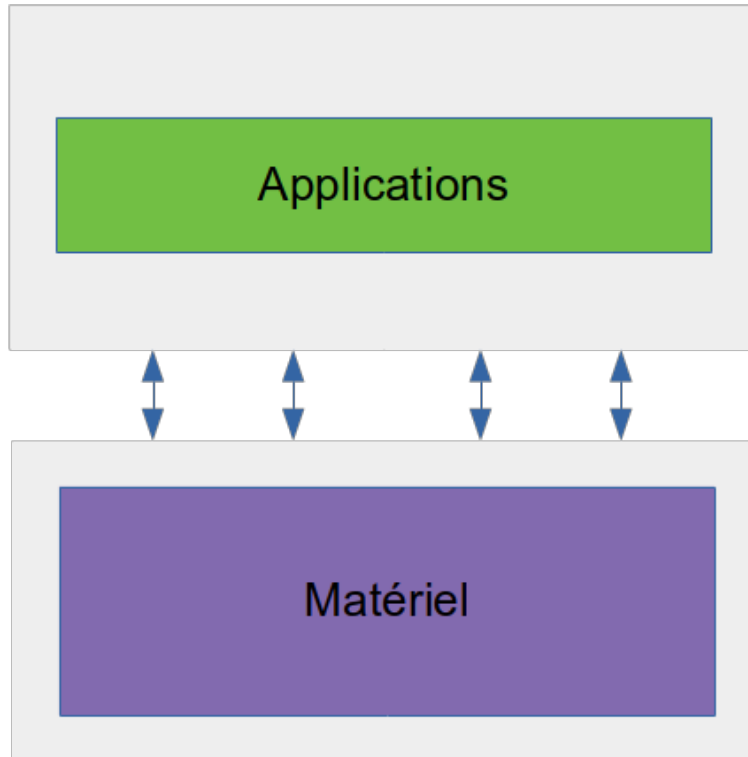
Pourquoi BSD : problème de licence. Si on utilisait GNU Linux, il y aurait obligation de laisser les codes sources sous GNU libres.

2.5 Exo-noyau



Le noyau est réduit à l'interception et redirection d'interruptions. Les applications ont la charge de la gestion de la mémoire et des périphériques.

2.6 Uni-Kernel



Une seule application en exécution dans le noyau. En vogue dans le domaine des systèmes embarqués. Ces uni-kernels sont ensuite encapsulées en VM, et chaque une possédant son propre espace mémoire.

3 Histoire de Linux

3.1 L'origine de "Unix"

Le nom UNIX fait référence au système d'exploitation MULTICS (MULTiplexed Information and Computing System) issu d'une collaboration entre le MIT, Bell & General Electric. Plusieurs rumeurs tournent autour de l'origine du mot.