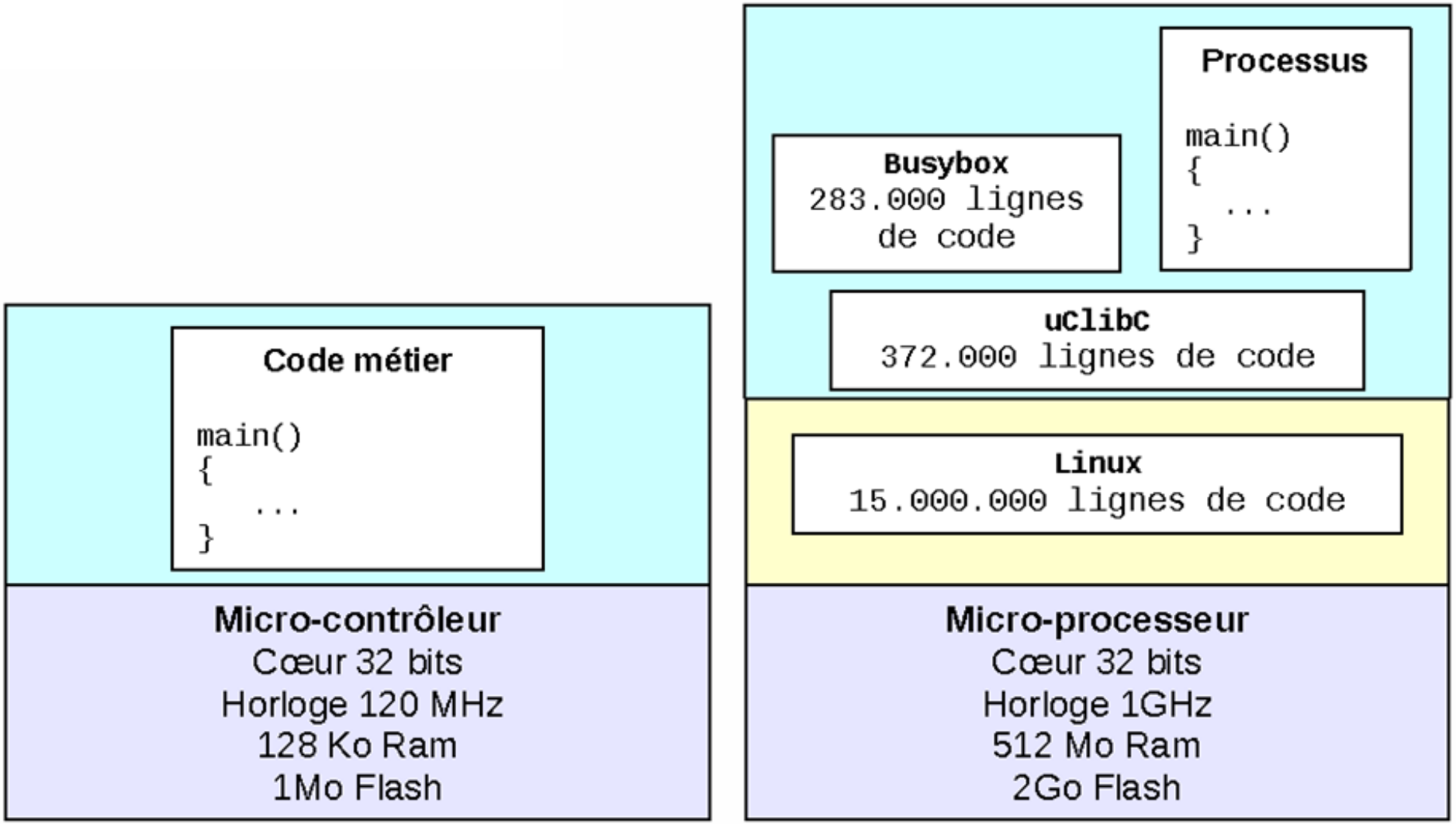
Systèmes d'exploitation et programmation Shell

Marc Silanus marc.silanus@univ-avignon.fr

Le système d'exploitation - Linux

comparaison des solutions matérielles



- Microcontroleur : faibles ressources matérielles, accès direct
- Microprocesseur : Matériel trop complexe pour une gestion directe

Définition d'un système d'exploitation

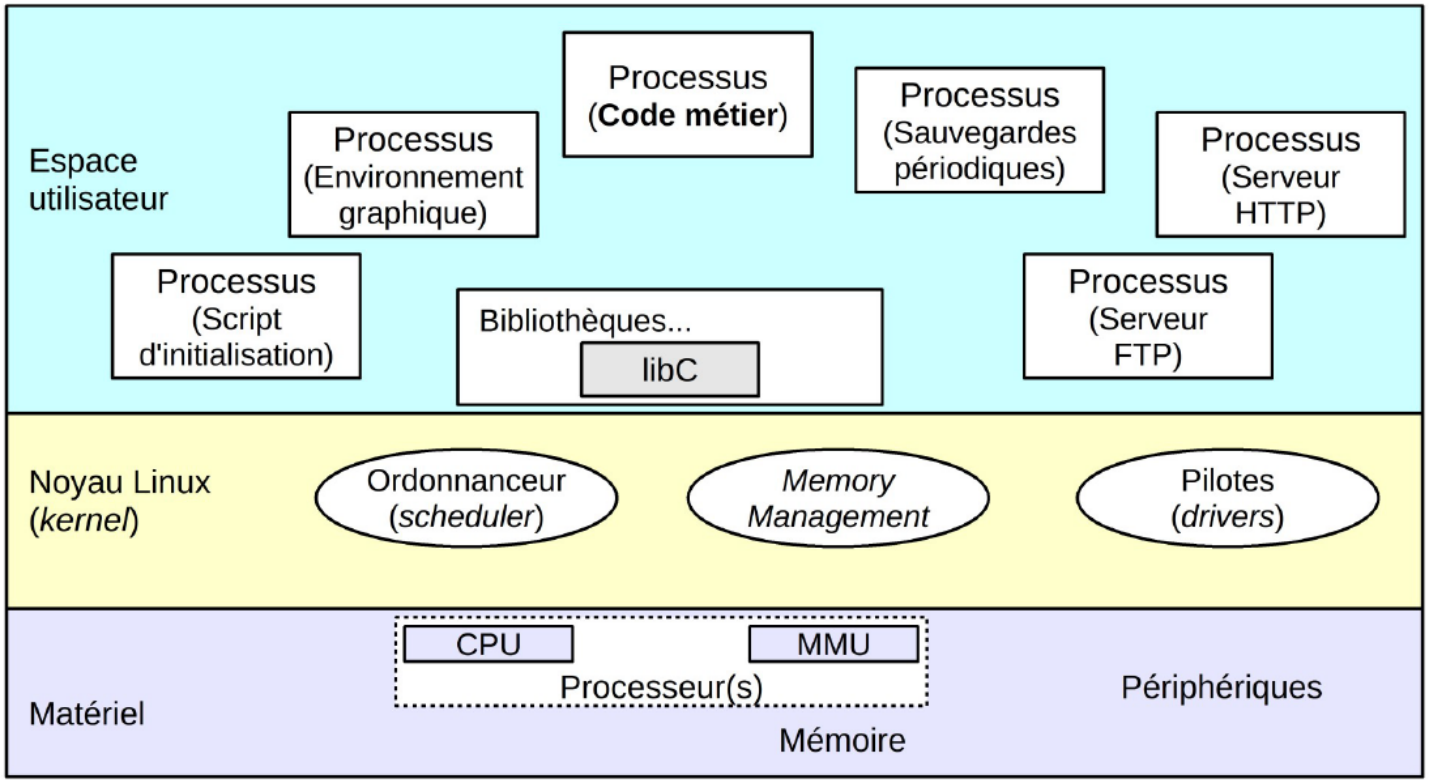
Pour l'utilisateur :

interface fournie avec l'ordinateur, le smartphone...

Plus général :

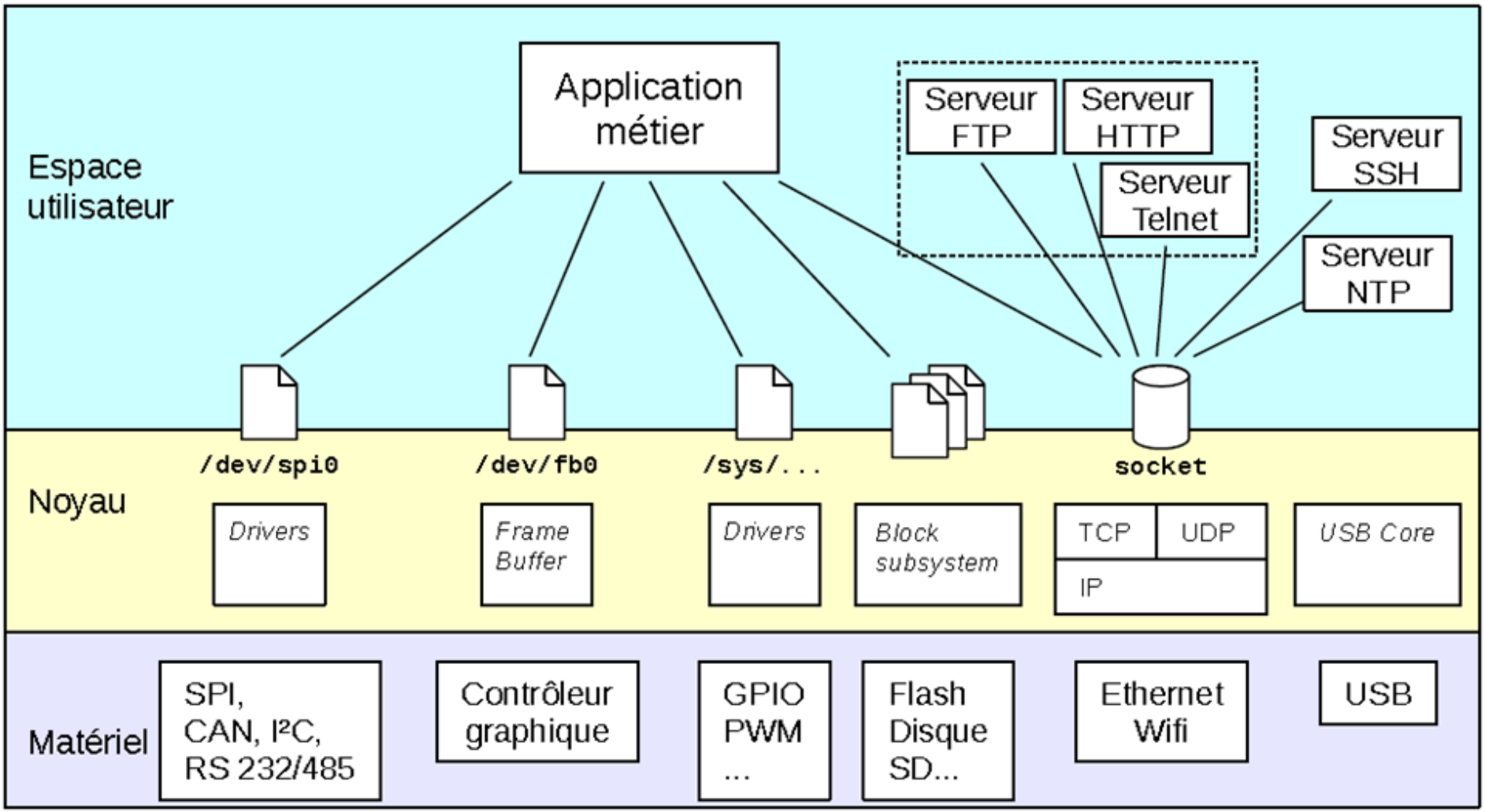
Logiciel qui gère le matériel et fournit un environnement pour les programmes applicatifs en exécution

Composant d'un OS : Linux



- Rôle du noyau Linux : mettre les ressources offertes par le matériel à disposition des applications de l'espace utilisateur.
- Lorsqu'il exécute le code du noyau, le processeur est en mode superviseur (privilégié).
- Pour exécuter du code en espace utilisateur, il passe en mode protégé (non-privilégié).

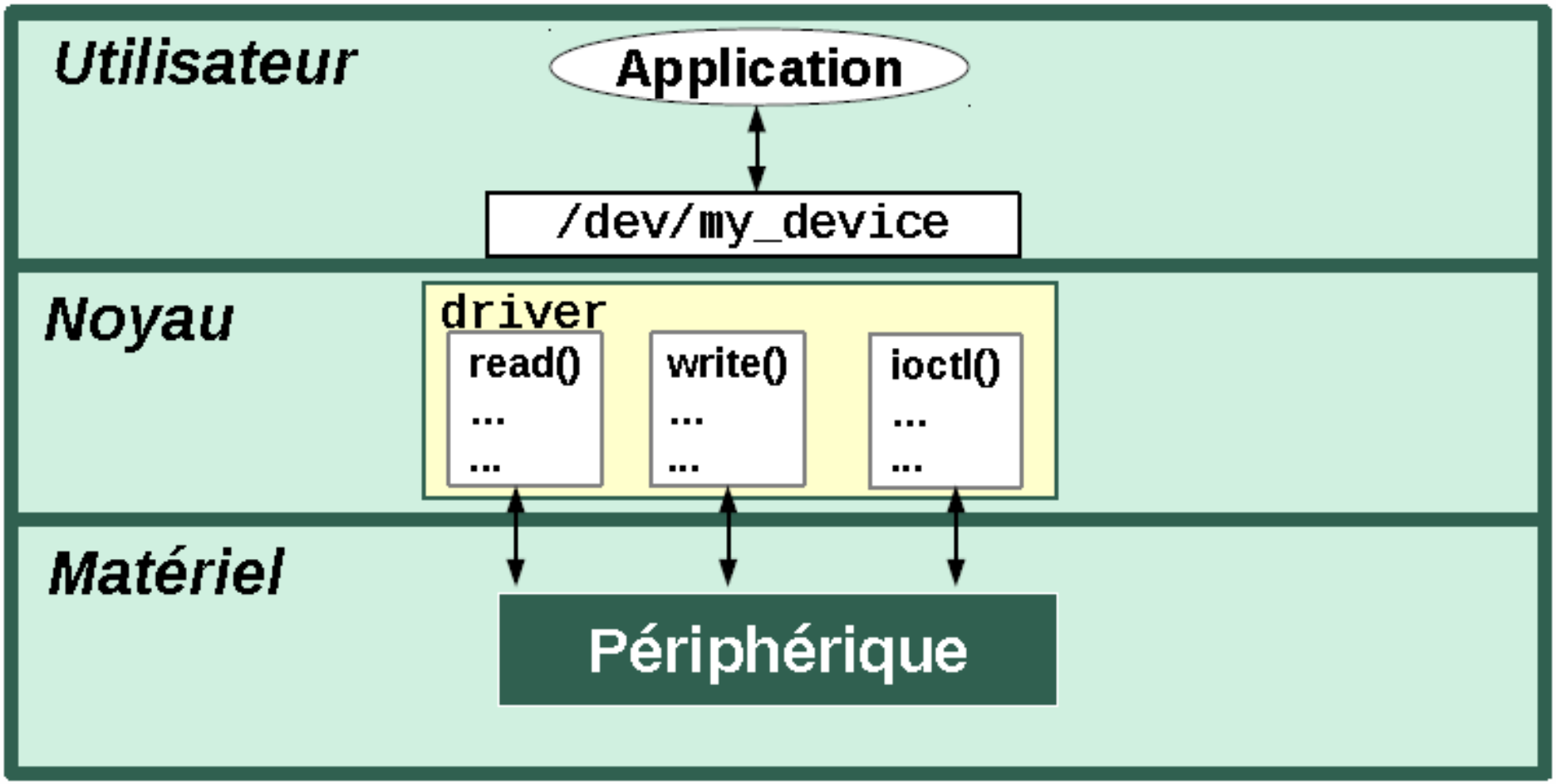
Abstraction des périphériques



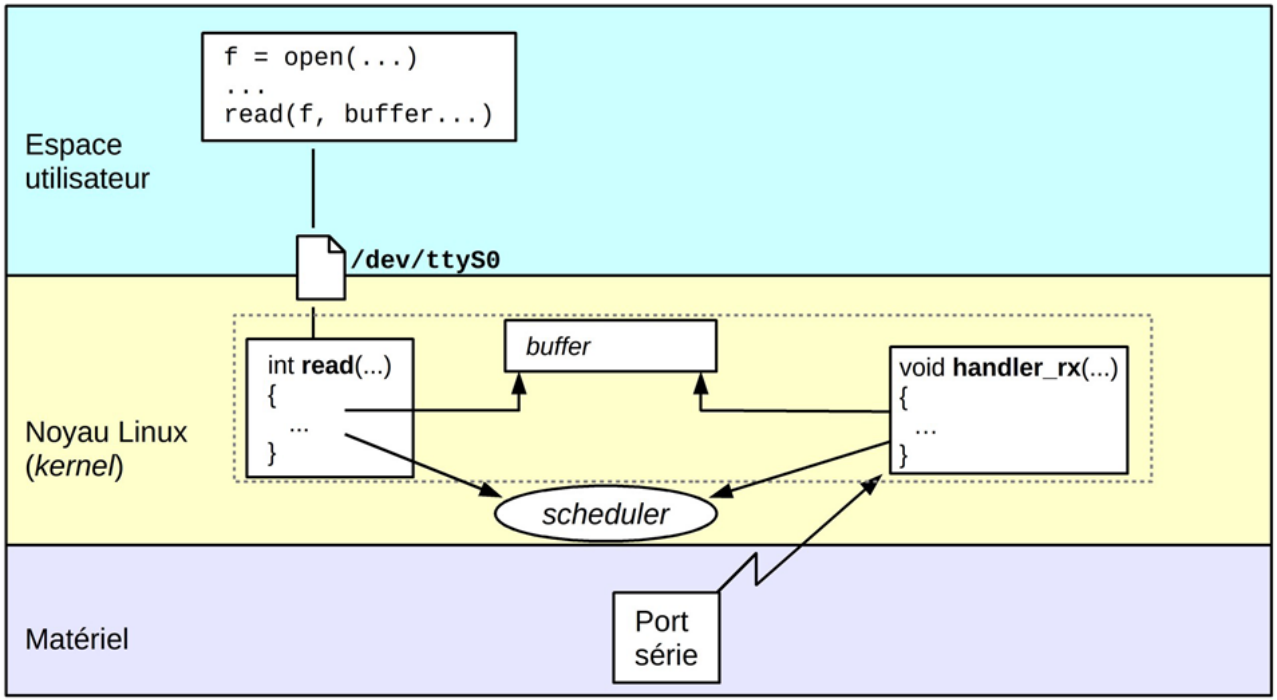
- Masquer la complexité matérielle
- Simplifer les accès au matériel

Abstraction des périphériques

Les applications de l'espace utilisateur communiquent avec les périphériques en réalisant des opérations (lecture, écriture paramétrage. . .) sur des pseudo-fichiers



Exemple : utilisation du port série

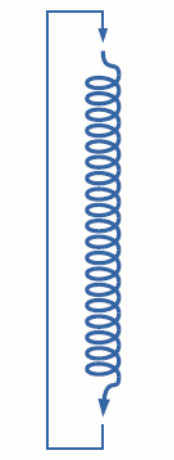


- Utilisateur : fonctions open() close() write() read() python par exemple
- Echange avec le matériel au moyen d'un pseudo-fichier /dev/ttySx
- Appel système par le noyau pour lire le buffer de réception
- Si des données sont présentes => lecture immédiate
- Sinon, invocation de l'ordonanceur pour endormir la tâche de lecture
- L'arrivée d'un caractère déclenche une interruptions matérielles
- L'ordonnanceur peut réveiller la tâche

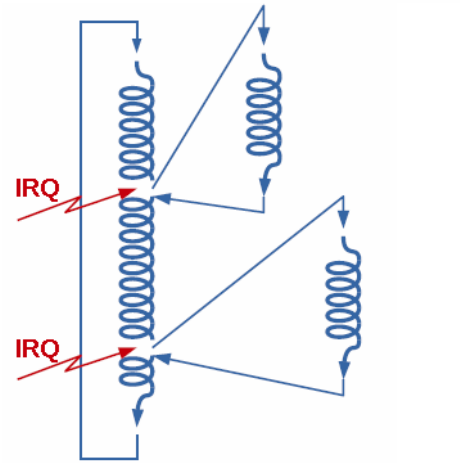
Exécution des tâches

Plusieurs stratégies :

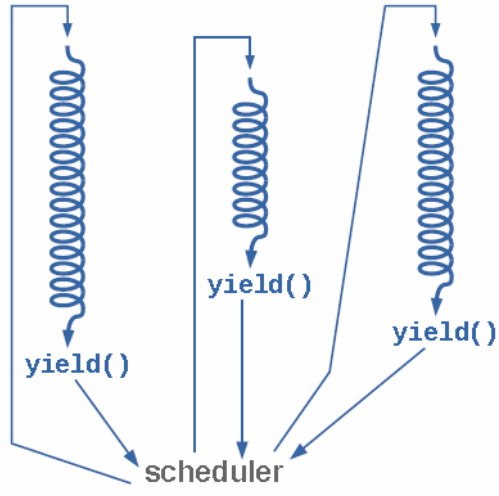
- Monotâges (comportement typique microcontroleurs/API)
- Multitâches : Nécessite un ordonnaceur



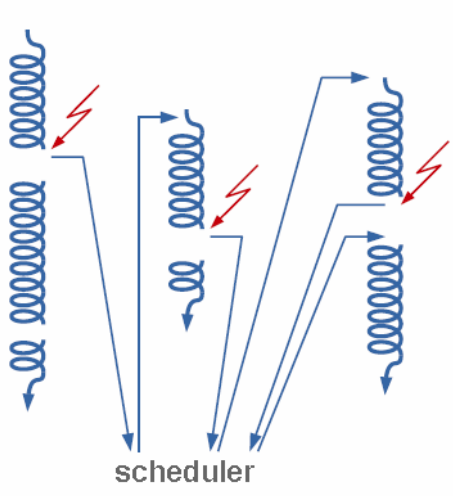
Superloop



Superloop avec interruptions



Coopératif



Préemptif

L'ordonnancement

Définition :

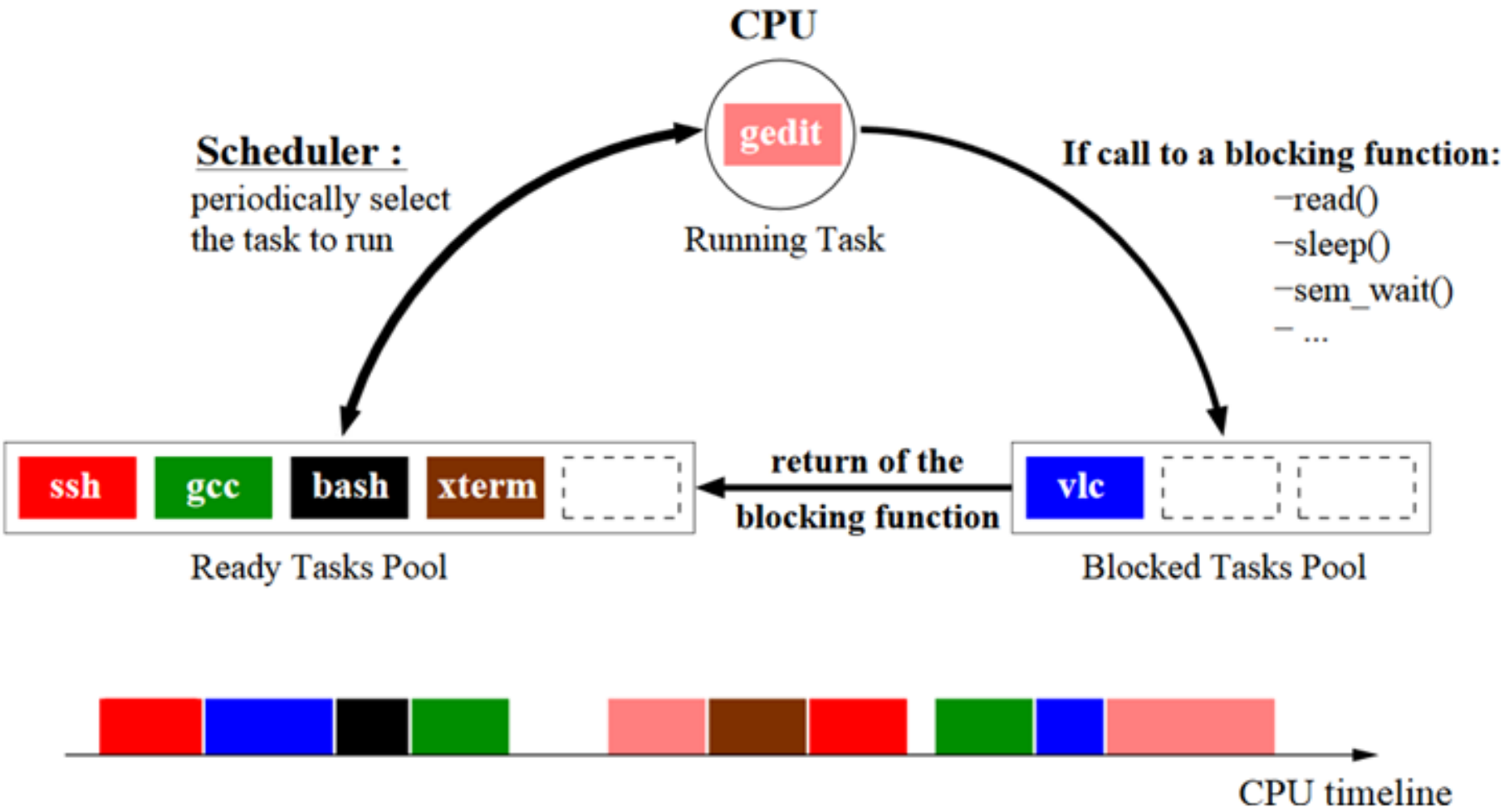
Tâche de sélection d'un processus en attente dans la liste des processus prêts et d'allocation du CPU pour ce processus

Il existe plusieurs modes d'ordonnancement :

- **Temps partagé (time sharing system)** : comportement par défaut sur les O.S. comme Linux
- **Temps réel (realtime scheduling)** : suivant des algorithmes comme Round Robin ou Fifo basés sur des priorités entre tâches ou Earliest Deadline First utilisant des temps d'expiration des tâches.

Etat des processus

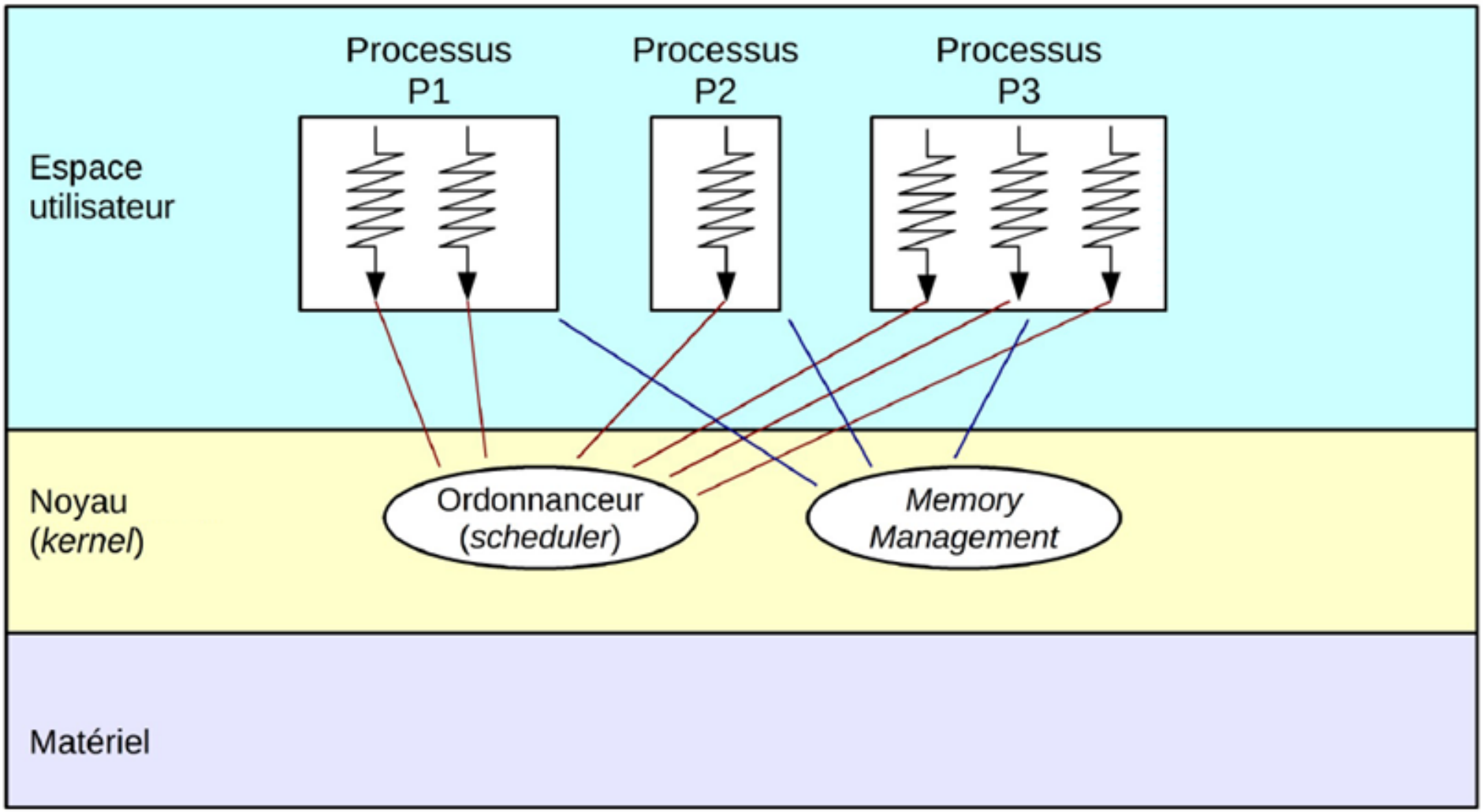
- scheduler(odonnanceur) : sélection du processus à exécuter
- CPU partage son temps entre plusieurs processus
- 3 états pour un processus : ready, running, blocked.



Processus et threads

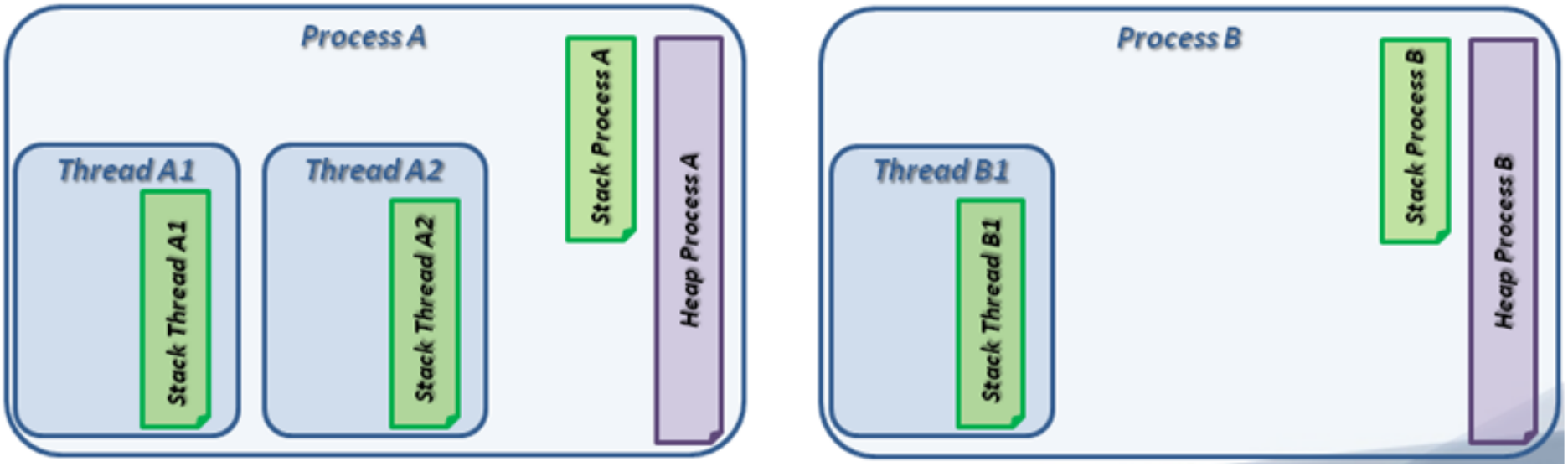
Definition :

Les processus sont des espaces de mémoire disjoints, au sein desquels s'exécutent un ou plusieurs threads

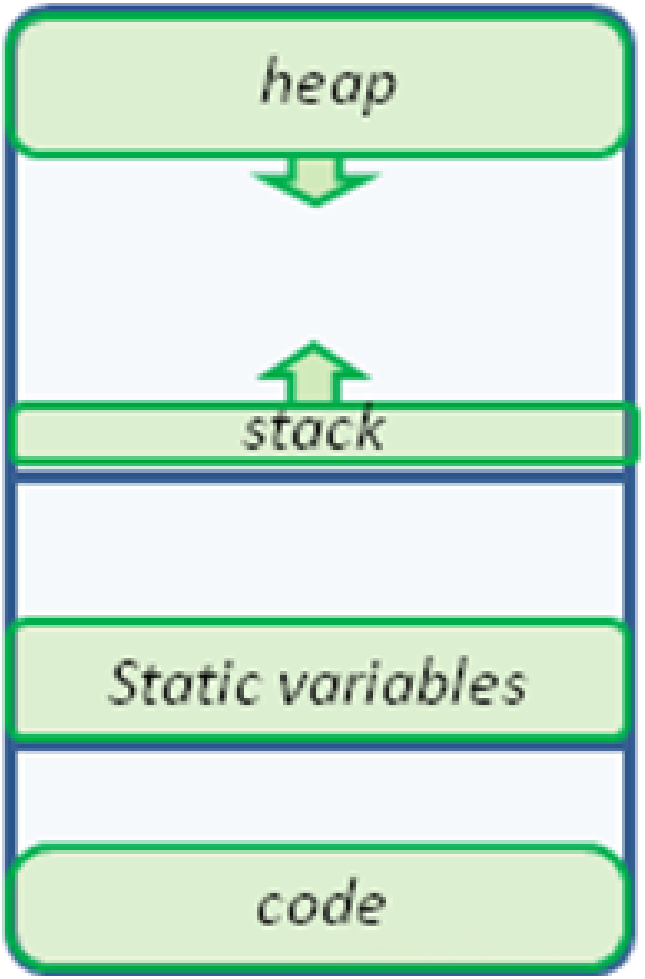


Processus et threads - la mémoire

- Le tas : commun à tout les threads d'un même processus
- La pile : chaque thread et processus possède sa propre pile

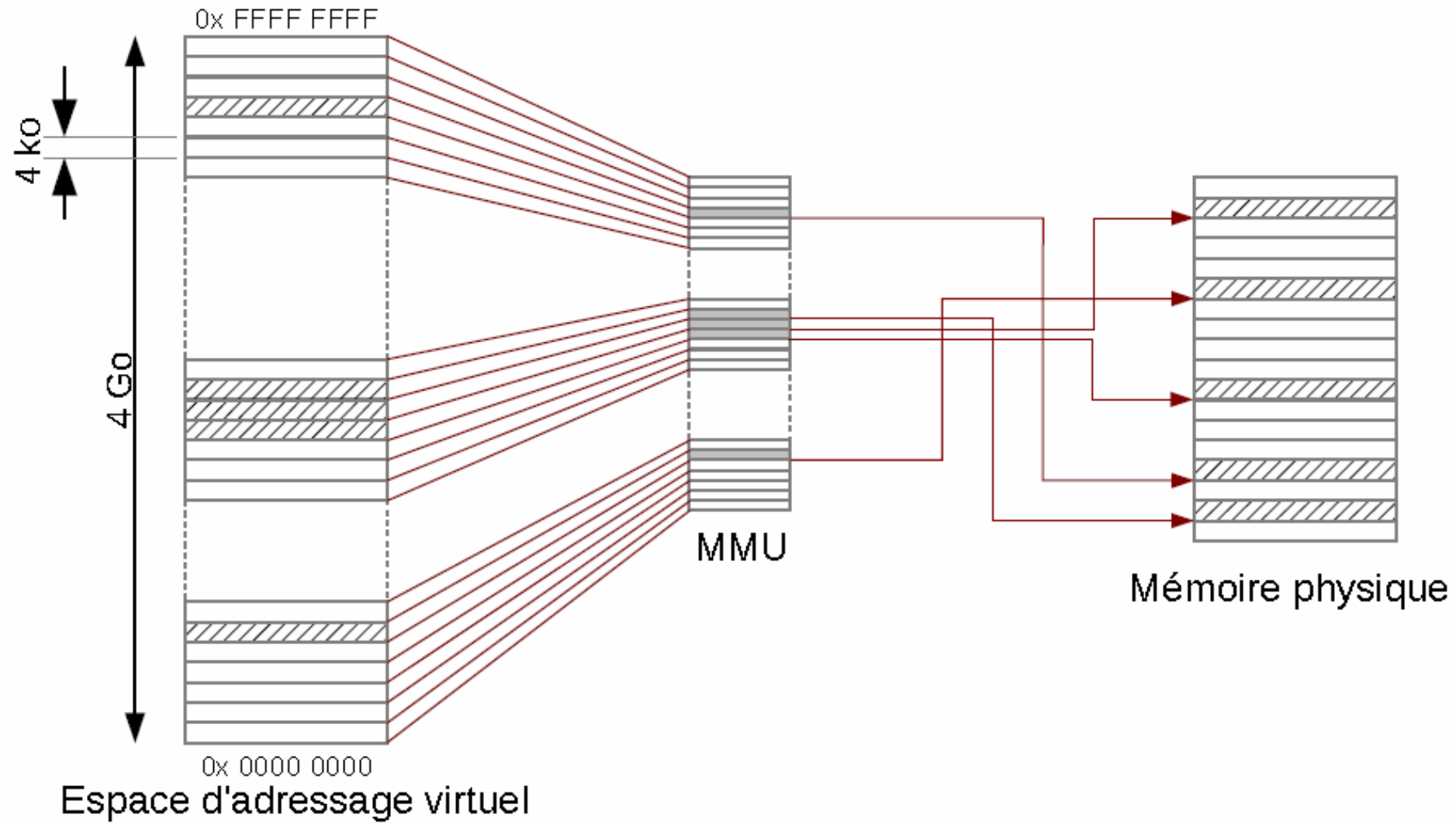


Processus et threads - la mémoire



- **Tas** : Allocations dynamiques. Python : variables globales et listes
- **Pile** : Variables locales, paramètres de fonctions, valeur et adresse de retour des fonctions

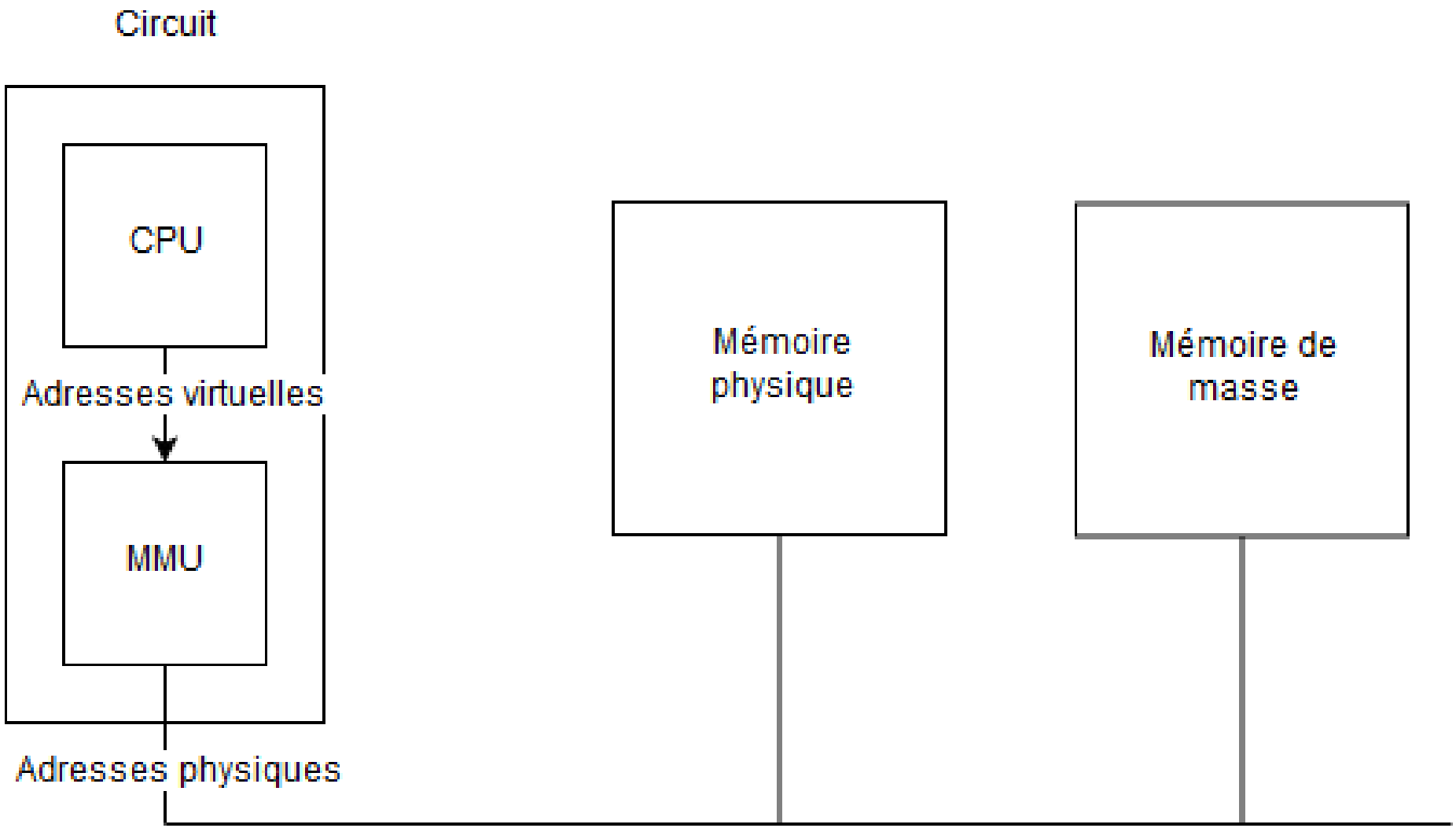
Mémoire virtuelle



Lorsqu'il est exécuté, un processus dispose de toute la mémoire disponible.

Oui mais c'est de la mémoire virtuelle !

Mémoire virtuelle



Mémoire virtuelle

Adresse virtuelle :

Adresse générée par la CPU et vue par le programme utilisateur

Adresse physique :

Adresse vue par l'unité de mémoire, c'est à dire chargée dans le registre d'adresse mémoire de la mémoire

Unité de gestion mémoire (MMU) :

Dispositif matériel intégré au microprocesseur associant les adresses logiques et physiques

Fonctions assurées par la MMU

Translation d'adresses :

- Segmentation : subdivision des espaces d'adressage d'après leur fonctionnalités.
- Pagination : subdivision des espaces d'adressage des tâches en petites tranches de taille fixe

Protection :

Chaque programme reste confiné dans son espace mémoire

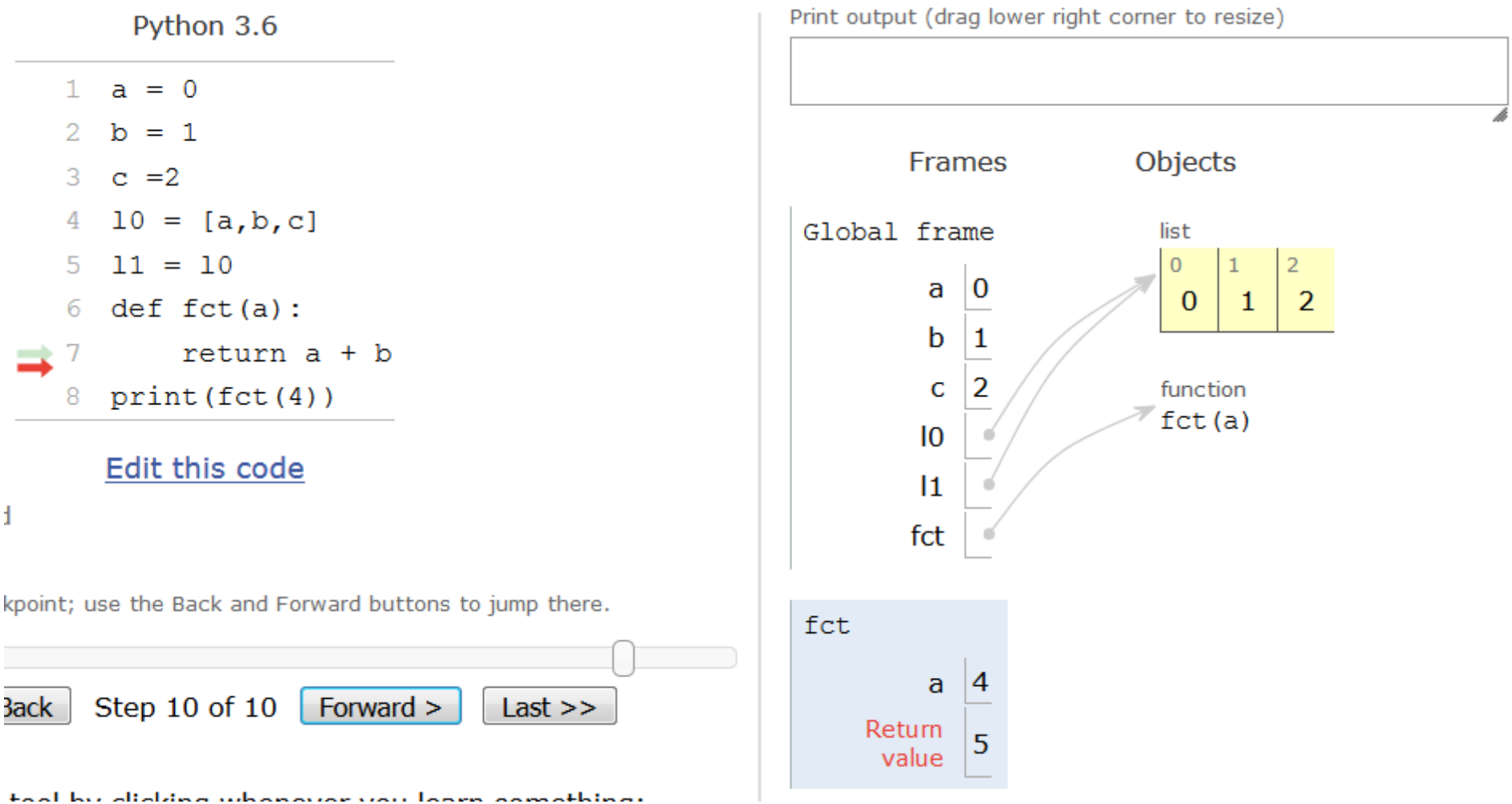
Mémoire virtuelle :

création d'un espace d'adressage important incluant la mémoire physique et une partie de la mémoire secondaire (>mémoire physique)

Swapping :

des portions de la mémoire sont rapatriées et envoyées vers l'espace de stockage secondaire.

Observation dans Python tutor



Python et la gestion de la mémoire

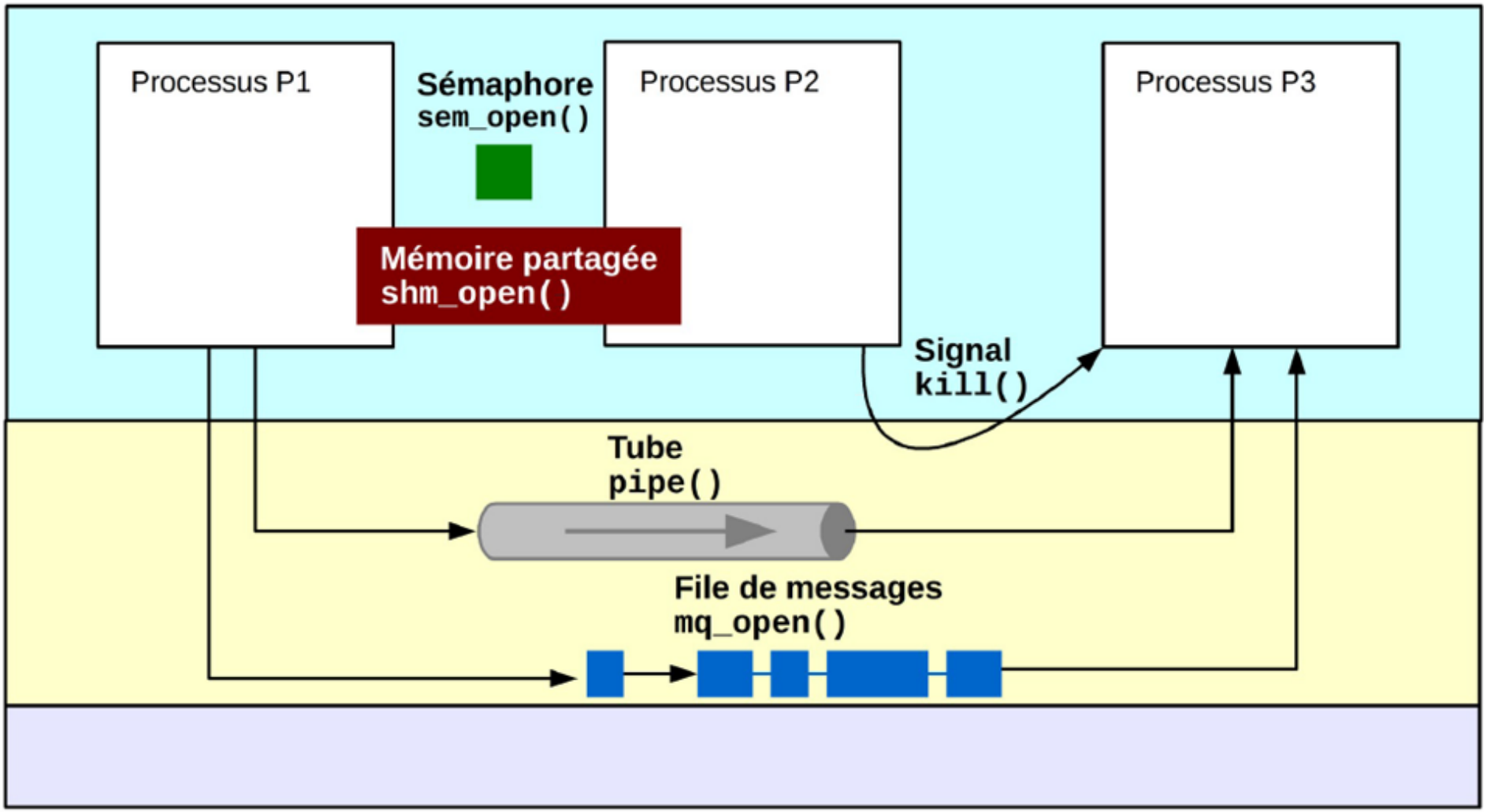
Adresse d'une variable

```
>>> a = 2
>>> hex(id(a))
>>> 0x5617d6cfe440
```

Contenu d'une adresse

```
>>> import ctypes
>>> ctypes.cast(0x5617d6cfe440, ctypes.py_object).value
>>> 2
```

Communication entre processus : IPC



Modularité du noyau Linux

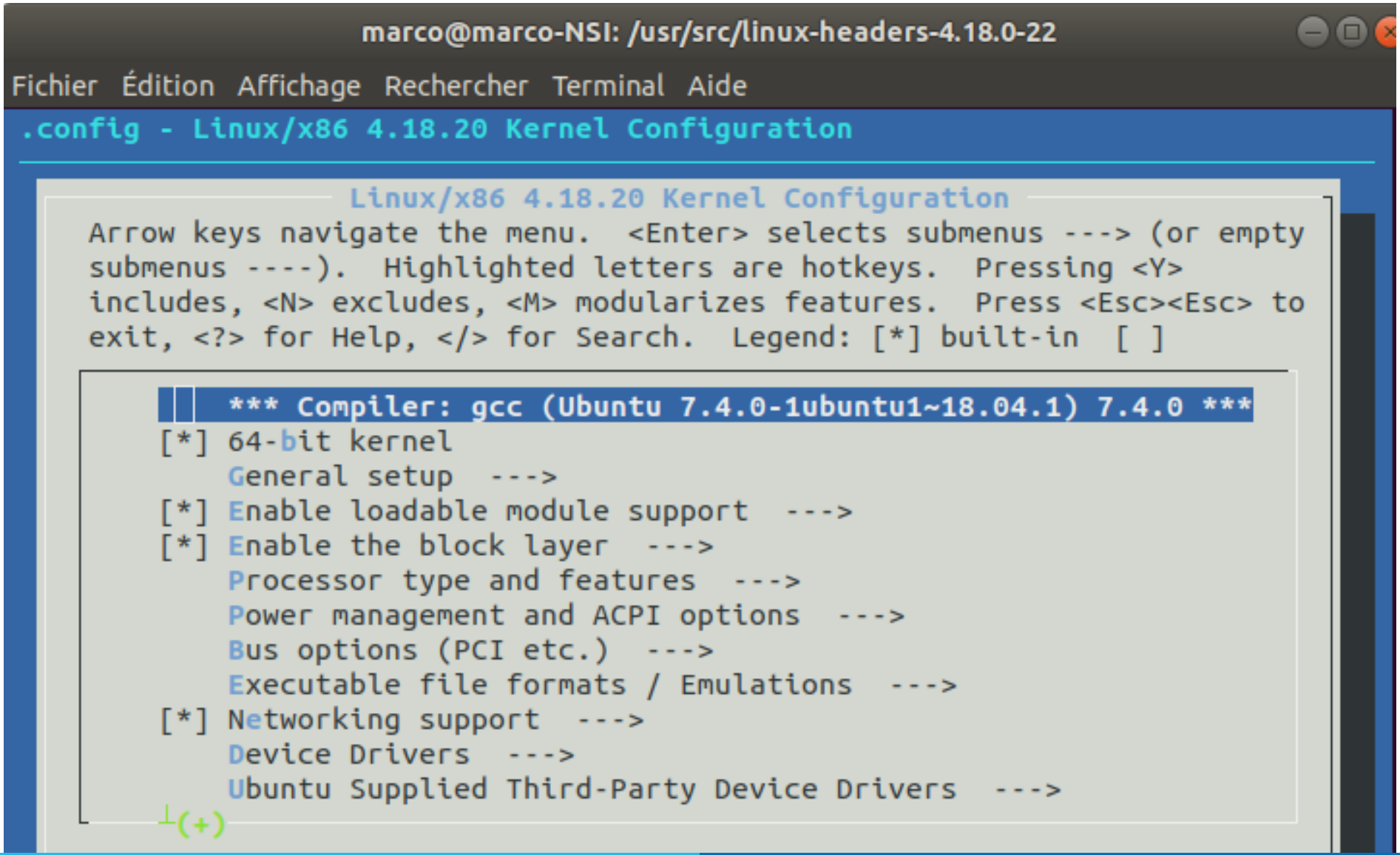
- Image du noyau (kernel image) : **un seul fichier** créé après édition de liens des différents fichiers objets
- Chargé en mémoire au démarrage.
- Disponibilité des fonctionnalités incluses dès le démarrage du noyau

Modules :

- compilation de certains éléments (pilotes de périphériques, systèmes de fichiers. . .)
- Chargement dynamique par le noyau en fonction des besoins
- Stockage de chaque module dans un fichier séparé (`*.ko` dans `/lib/modules`)
- Pas d'accès lors du démarrage initial
- Allègement du noyau, durée de la séquence de boot réduite

Configuration et compilation du noyau

`make menuconfig`



Configuration et compilation du noyau

```
make menuconfig
make
```

Make :

- A saisir dans le répertoire racine des sources du noyau
- Option `-j <n>` pour accélérer la compilation en utilisant plusieurs coeurs du processeur

Génère :

- `vmlinux` : image non compressée du noyau mais non bootable
- `arch/<arch>/boot/*Image` : image finale et bootable, généralement compressée, du noyau (`bzImage` pour **x86**, `zImage` pour **ARM**...)
- Tous les modules du noyau, répartis dans l'arborescence des sources, sous la forme de fichiers `*.ko`

Shell

Shell :

Couche logicielle qui fournit l'interface utilisateur d'un système d'exploitation

Le shell d'un système d'exploitation peut prendre deux formes distinctes :

- Interface en ligne de commande (CLI)
- Shell graphique fournissant une interface graphique pour l'utilisateur (GUI, pour Graphical User Interface).

Interpréteur de commande

- Tout utilisateur est associé à un type de shell (`/bin/bash`)
- Le prompt indique :
  - le nom d'utilisateur
  - le nom de la machine
  - la localisation dans le système de fichier
  - les privilèges (superutilisateur : #, utilisateur : $)
- commandes : fichiers executables généralement siutés dans `/bin`
- Accès au manuel : `man`
- Aide sur une commande : `help <commande>`

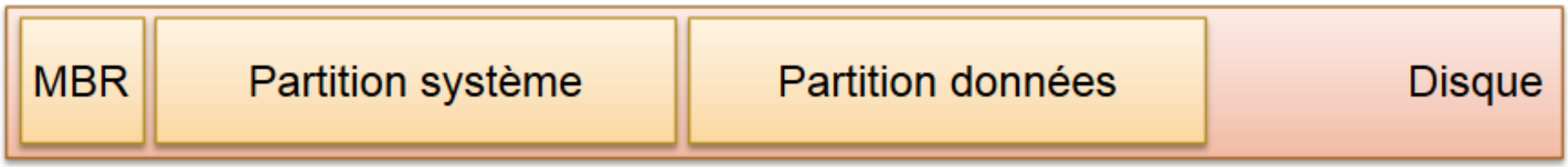
Shell

Exécutions de plusieurs script au démarrage pour définir l'environnement de l'utilisateur :

- `/etc/profile` : commun tous les utilisateurs
- `/etc/profile.d/*.sh` : si le dossier existe
- `$HOME/.profile` : profil personnel de l'utilisateur
- `$HOME/.bashrc` : fonctions et alias personnel. Exécuté à chaque ouverture d'un terminal.
- `/etc/bash.bashrc` : alias globaux et définition du prompt $P1

Système de fichiers

- Support matériel : Disque dur, flash, SDcard, microSD, ...
- Organisation matérielle minimum :



- Le Master Boot Record est situé dans les 1er secteurs du disque
- Il est constitué de 2 parties :
  - La table des partitions
  - Le programme d'amorçage qui charge le noyau du système
- Partition système contient :
  - L'image du noyau
  - Les fichiers nécessaires au lancement du système
- La partition données contient le systèmes de fichiers

Système de fichiers

Chaque système est associé à un format de données

- Sous Linux
  - ext2, ext3, ext4, jfs, xfs, ...
  - ext2 non journalisé
- Sous Windows
  - fat, fat32, ntfs

Toujours préférer un système de fichier « journalisé »

- Chaque séquence de lecture/écriture est d'abord inscrite dans un journal avant d'être effectuée
- Si le système se bloque pendant la séquence, elle sera achevée après le redémarrage

Système de fichiers

- `/dev` : Fichiers spéciaux de lien avec les périphériques matériels

Types de bus :

- `hd` : Périphériques IDE
- `sc` : Périphériques SCSI
- `sd` : Périphériques SATA

Exemples :

- `/dev/hda1` : Partition 1 sur le 1er disque IDE
- `/dev/sdb2` : Partition 2 sur le 2ème disque Sata

Système de fichiers

Rep	Description
`/`	Répertoire “racine”
`/boot`	Contient le noyau Linux et l'amorceur
`/bin`	Contient les exécutables de base, `cp`, `mv`, `ls`, ...
`/dev`	Contient des fichiers spéciaux liés aux périphériques matériels
`/etc`	Contient les fichiers de configuration du système
`/home`	Contient les fichiers personnels des utilisateurs
`/lib`	Contient les librairies et les modules du noyau (`/lib/modules`)
`/media`	Contient les points de montage des lecteurs cd, dvd, clef usb, ...
`/proc`	Contient une “image” du système : instantané de son état
`/root`	Répertoire personnel du superutilisateur
`/sbin`	Contient les exécutables destinés à l'administration du système
`/tmp`	Contient des fichiers temporaires utilisés par certains programmes
`/usr`	Contient les exécutables des programmes, les sources de linux, ...
`/var`	Contient les fichiers de maintenance du système (`log`)

Système de fichiers

Se déplacer dans les dossiers

```
$ cd
$ cd ~/depuis/mon/dossier/perso
$ cd /chemin/absolu
$ cd chemin/relatif
$ cd ..
```

Système de fichiers

Lister le contenu d'un dossier

```
$ ls
```

Avec toutes les infos

```
$ ls -l
```

les fichiers cachés aussi

```
$ ls -al
```

Système de fichiers

Créer un dossier

```
$ mkdir nom_dossier
```

Supprimer un dossier vide

```
$ rmdir nom_dossier
```

Supprimer un dossier et tout ce qu'il contient

```
$ rm -r nom_dossier
```

Système de fichiers

Déplacer/renommer

```
$ mv dossier_src dossier_dest
```

copier un dossier vide

```
$ cp dossier_src dossier_dest
```

copier un dossier non vide

```
$ cp -r dossier_src dossier_dest
```

Système de fichiers

Tout fichier appartient à un propriétaire unique et à un groupe d'utilisateurs

```
$ -rw-r--r--    1   user user 445  juin  12 11:22 ieee754.py
$ drwx------    3   root root 4096 févr. 10 01:18 root
$ crw-rw----    1   root dialout 4, 65  juin 21 15:06 ttyS0
```

type de fichier : "Sous Unix tout est fichier"

- `b` : pseudo fichier périphérique binaire
- `c` : pseudo fichier périphérique caractères
- `d` : dossier

Autorisations

- `r` : autorisé à lire
- `w` : autorisé à écrire
- `x` : autorisé à exécuter

Système de fichiers

commandes associées

```
$ chown propriétaire fichier
$ chgrp propriétaire fichier
$ chmod +x fichier
$ chmod 755 fichier
```

Système de fichiers

Créer un fichier

```
$ echo "du texte" > fichier
$ echo "du texte" >> fichier
$ vi fichier
$ nano fichier
$ gedit fichier
```

lire un fichier

```
$ cat fichier
$ less fichier
$ vi fichier
$ nano fichier
$ gedit fichier
```

Système de fichiers

Déplacer/renommer

```
$ mv fichier_src fichier_dest
```

Copier

```
$ cp fichier_src fichier_dest
```

Supprimer

```
$ rm fichier
```

Rechercher un fichier

```
$ find /depuis -name fichier
```

Rechercher dans un fichier

```
$ grep a_trouver dans_fichier
```

Interaction

```
$ echo "Hello word !"
Hello word !

$ read ma_var
n'importe quoi
$ echo $ma_var
n'importe quoi
```

Programmation Shell

Script bash

- fichier texte contenant une suite de commandes shell
- s'exécute comme une commande unique
- peut définir et utiliser des variables
- on peut lui passer des paramètres
- gère les entrées/sorties et les redirections
- possède des structures conditionnelles et itératives
- on peut définir des fonctions internes

Véritable langage de programmation interpété

Exemple

```
#!/bin/bash
# script bonjour
# affiche un salut à l'utilisateur qui l'a lancé
# variable d'environnement $USER : nom de login
echo ---- Bonjour $USER -----
# l'option -n empêche le passage à la ligne
# le ; sert de séparateur des commandes sur la ligne
echo -n "Nous sommes le " ; date
# recherche $USER en début de ligne dans passwd
# puis extraction de l'uid au 3ème champ, et affichage
echo "Ton numéro d'utilisateur est : "\
$(grep "$USER" /etc/passwd | cut -d: -f3)
```

Programmation Shell

exécution

```
$ bash script
```

Avec les bons droits :

```
$ chmod u+x script
$ ./script
```

En faire une commande :

```
$ mv script ~/bin # Créer le dossier avant !
$ export PATH=~/bin:$PATH # pour le shell courant
$ echo PATH=~/bin:$PATH >> $HOME/.bashrc # permanent
```

Crédit images

Christophe Blaess :

Ingéniérie et formation sur les sytèmes libres

http://https://www.blaess.fr/christophe/