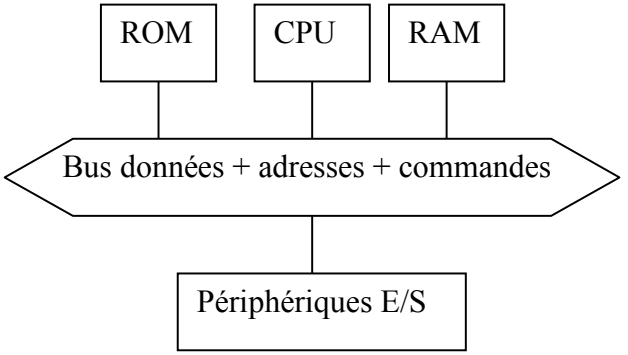
Programmation système et réseau

Rappel : architecture d'un ordinateur



- CPU : Central Processing Unit
- ROM : Read-Only Memory
- RAM : Random Access Memory
- Périph : disques, cartes réseau, graphique, clés usb, imprimantes, ...

CPU exécute des instructions simples
Cycle :
• rechercher une instruction dans la RAM
• éxécuter l'instruction

OS : une machine virtuelle

Avant OS (*Operating System*) :

- machine = CPU + cases mémoires (RAM, registres des contrôleurs)
- programmes dépendants de la machine

Avec un OS :

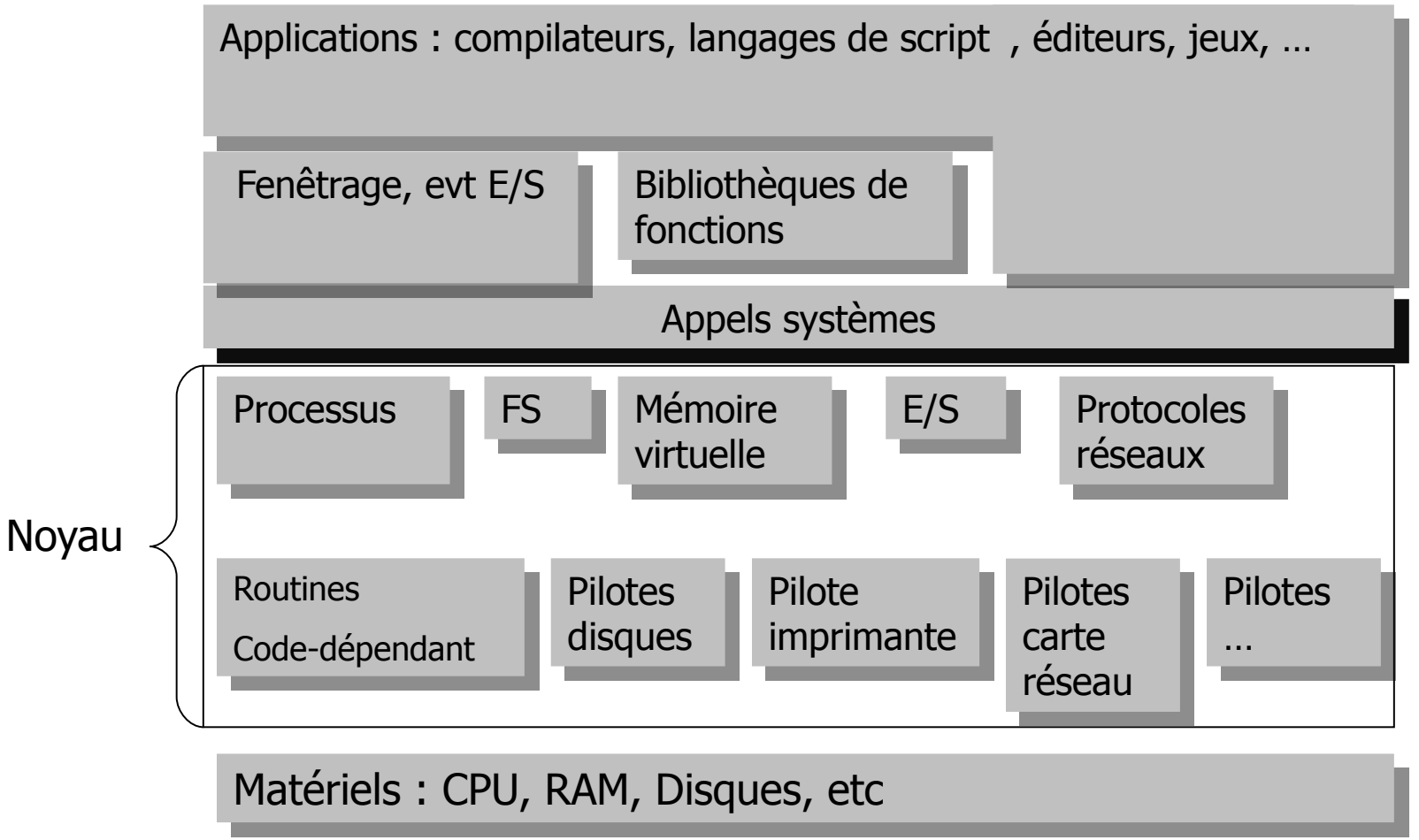
- machine = ensemble de fichiers, de programmes
- programmes peuvent être écrit dans un langage indépendant de l'architecture

Rôle d'un OS

Gestion :

- fichiers
- processus
- mémoire
- utilisateurs
- périphériques
- réseau

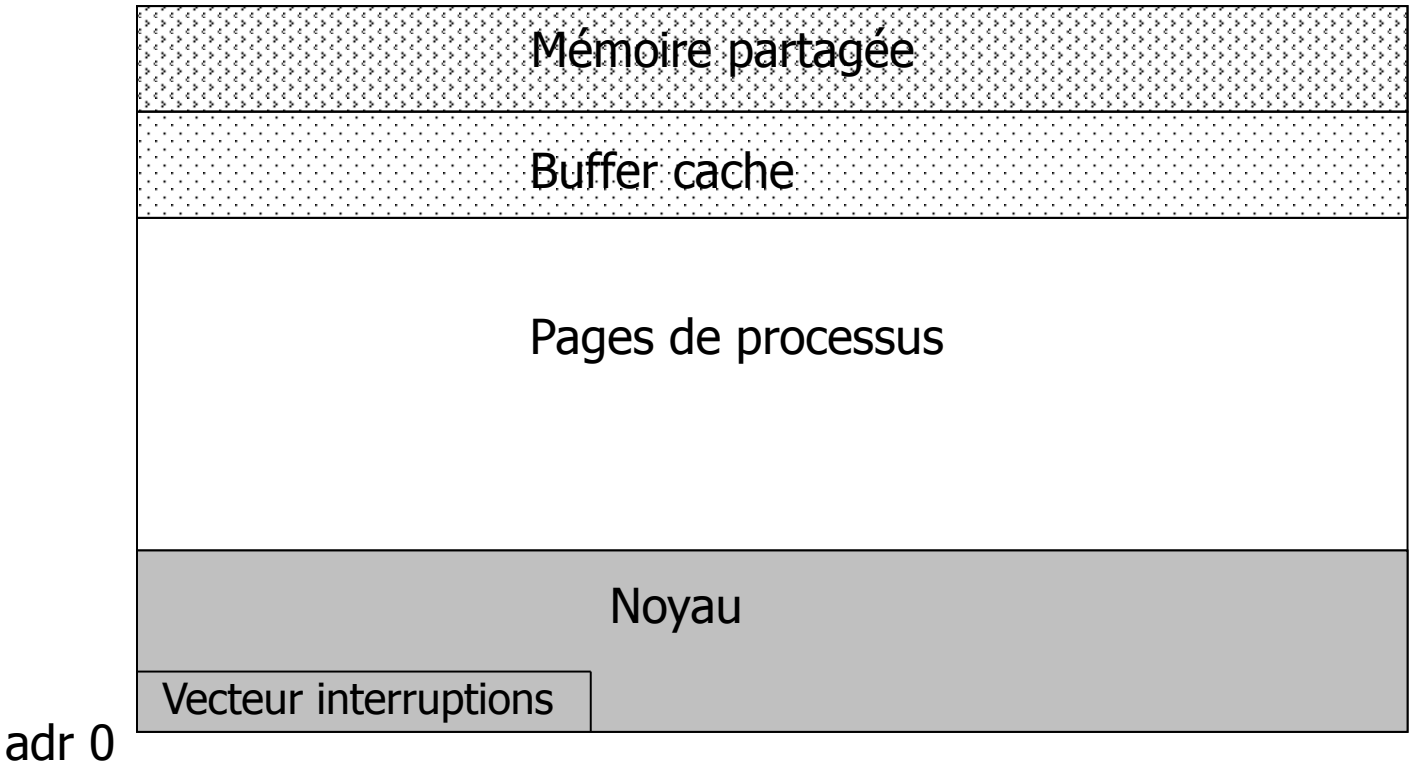
Structure d'un OS



OS : ensemble de fonctions

- http://www.makelinux.net/kernel_map
- quelques primitives remarquables :
  - drivers
  - routines d'interruptions
  - appels systèmes

Appels système : mécanisme



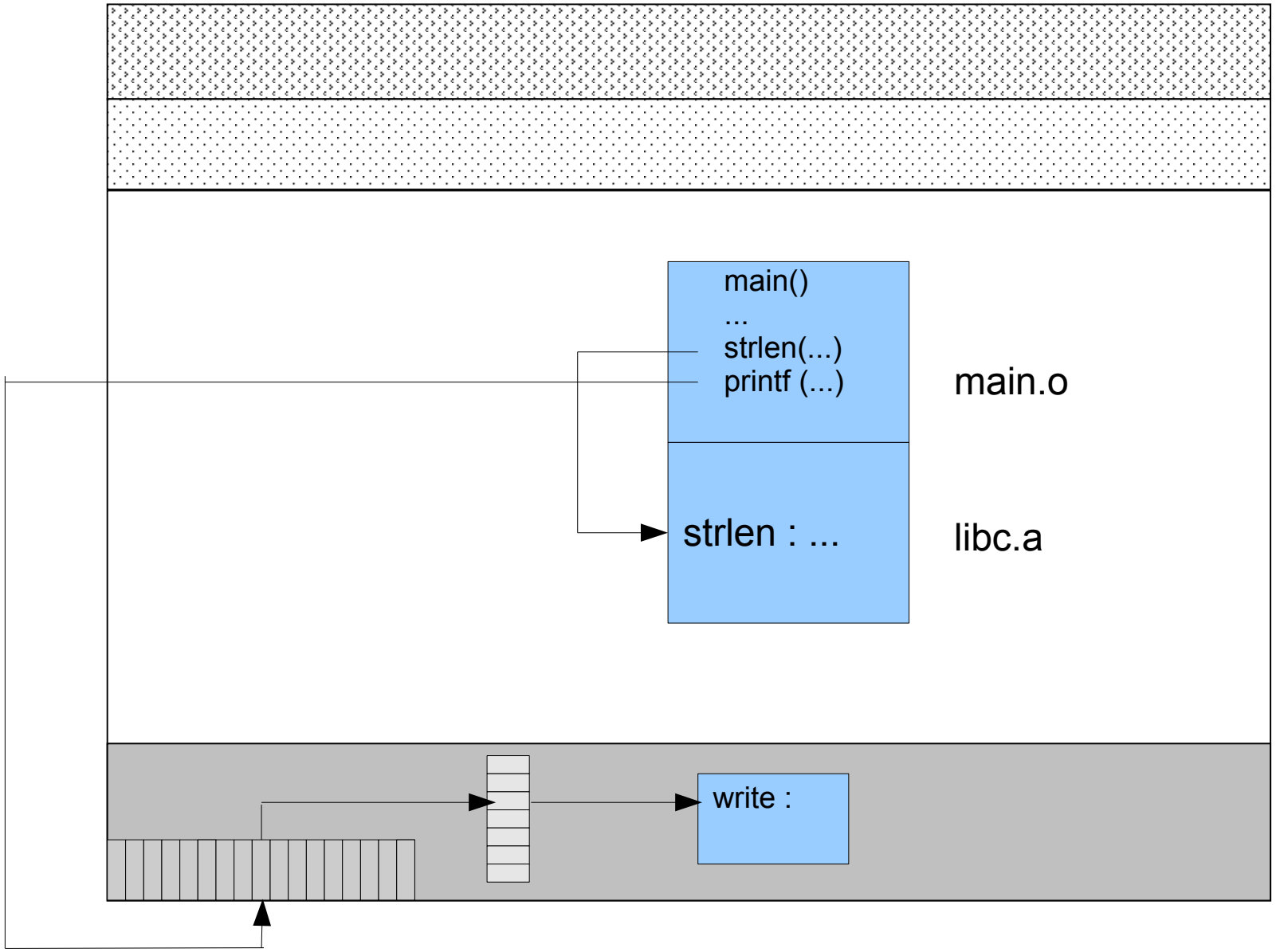
Appels système : mécanisme

main.c

```
#include <stdio.h>
#include <string.h>


main() {
    char *s = "Bonjour à tous\n ";
    int nb = strlen(s);
    printf(" %s contient %d lettres\n", s, nb);
}
```

Appels système : mécanisme



Appels système : mécanisme

- L'appel à une primitive système est très couteux par rapport à un simple appel de fonction (bibliothèque)
- Exécution :
  - Le cpu « sauve son état » (pointeur de programme, de pile, etc.)
  - Le cpu passe en mode privilégié
  - Le noyau récupère et vérifie les paramètres
  - Exécution de la primitive
  - Restauration des registres CPU

Appels système

1. Process Control.
    * load, execute
    * create process, terminate process
    * get/set process attributes
    * wait for time, wait event, signal event
    * allocate, free memory
2. File management.
    * create file, delete file
    * open, close
    * read, write, reposition
    * get/set file attributes
3. Device Management.
    * request device, release device
    * read, write, reposition
    * get/set device attributes
    * logically attach or detach devices
4. Information Maintenance.
    * get/set time or date
    * get/set system data
    * get/set process, file, or device attributes
5. Communication.
    * create, delete communication connection
    * send, receive messages
    * transfer status information
    * attach or detach remote devices

En résumé

- processus accède à son espace d'adressage
- pour accéder à des fichiers (R/W sur disque), aux terminaux (écran), mémoire (mémoire partagée), poser un verrou sur un fichier => faire appel au système
- appel système = code dans le noyau s'exécutant en mode privilégié (tout le jeu d'instruction)
- si appel système : interruption du processus pour que le noyau prenne la main
- la plupart des appels système rendent un entier : code d'erreur
  - 0 si ok
  - < 0 si échec (affichage avec perror)