Introduction et rappels

INF3173 – Principes des systèmes d'exploitation
Automne 2024

Francis Giraldeau
francis.giraldeau@uqam.ca

Université du Québec à Montréal

Agenda

- Modalités du cours INF3173
- Rôle d’un système d’exploitation
- Survol des concepts

À propos de votre professeur

- Francis Giraldeau
  - Bac en génie électrique, UdeS
  - Maîtrise en sciences informatique, UdeS
  - Doctorat en génie informatique, Polymtl
  - Simulation numérique, CNRC
  - Nouveau professeur à l'UQAM
- Courriel: giraldeau.francis@uqam.ca
- Bureau PK-4820
- Disponibilité mercredi (avec rendez-vous)

Plan de cours

- 1 intro
- 2 processus
- 3 fichiers
- 4 ordonnaceur
- 5 mémoire
- 6 stockage
- 7 exercices
- 8 examen intra
- 9 communication
- 10 synchronisation
- 11 interblocage
- 12 asynchrone
- 13 Win32, VM
- 14 exercices
- 15 examen final

Évaluations

- Quiz 1 (5%)
- Quiz 2 (5%)
- TP1 (10%)
- TP2 (15%)
- TP3 (15%)
- Examen mi-session (25%)
- Examen final (25%)

Modalités

- Matériel pour l’examen:
  - Feuille résumée format lettre, recto-verso
- TP indivituel ou en équipe de deux
  - Chaque étudiant doit maîtriser la matière
  - Sujet à question aux examens
  - Retard avec pénalité
  - Retour en classe
- Note minimale de 50% aux examens requis et 50% globalement pour réussir le cours
- Discussion: utilisation acceptable de ChatGPT

Livres de référence

- The Linux Programming Interface, Michael Kerrisk
- Modern Operating System, Tanenbaum
- Operating System Concepts, Silberschatz, Galvin, Gagne
- Understanding the Linux Kernel, Bovet, Cesati

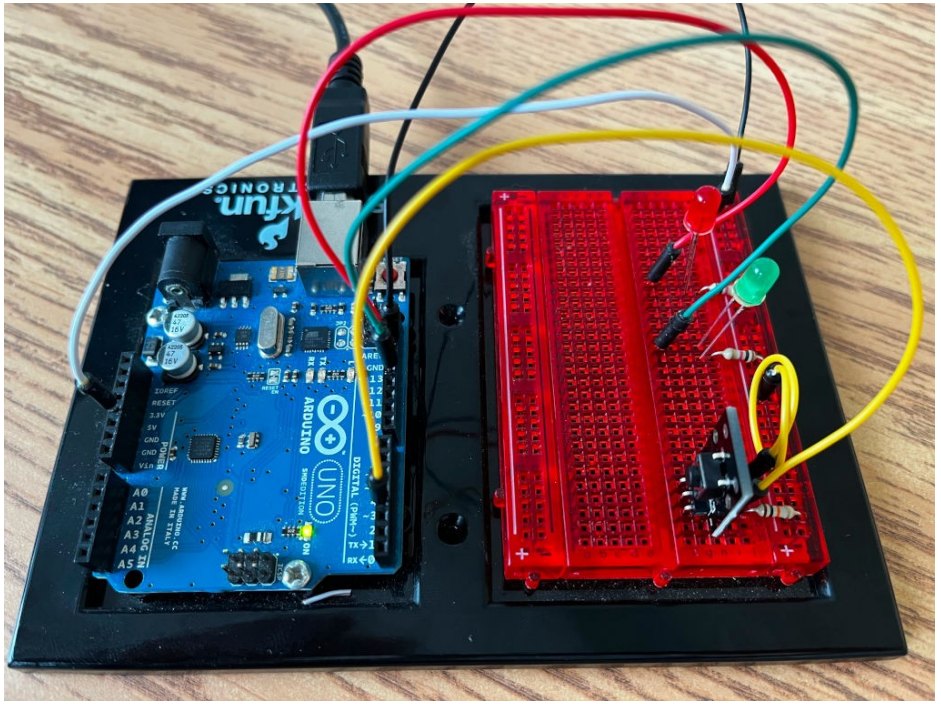
Ressources

- Diapos et liens sur Moodle
- Ordinateur avec Linux ou VM Linux
- Recommandé: Ubuntu 24.04
- IDE recommandé: QtCreator
  - Vous devez être en mesure de déboguer vos programmes en C
- TraceCompass
- Accès Gitlab (inf3173-scratchpad)
- Canal TEAMS

Excellence

- Le cours est difficile et nécessite beaucoup d’effort
- Mon engagement: vous apporter tout le soutient nécessaire pour votre réussite
- Votre responsabilité: essayer par vous-même, lire la documentation, voir les erreurs comme une occasion d’apprendre
- Intégrité: l’entraide est encouragée, mais aucun plagiat ne sera toléré

Un monde sans SÉ?



- Arduino UNO
- Processeur: Atmel Atmega328p
- Instructions AVR
- Pin 13 output LED rouge
- Pin 12 output LED verte
- Pin 3 input bouton

Exemples

- MultiTask: clignotement avec delay()
- MultiTaskAsync: clignotement par scrutation
- TaskSwitch: changement de la tâche active par une interruption
- ScanMemory: affichage de la mémoire
- BogusTask: erreurs fatales

Un mode sans SÉ?

- Instable: une tâche peut crasher le système
- Fragile: une tâche peut monopoliser le CPU et la mémoire
- Insécure: mémoire globale, pas de permission
- Fuites: impossible de récupérer la mémoire
- Conflit d’accès aux ressources partagées

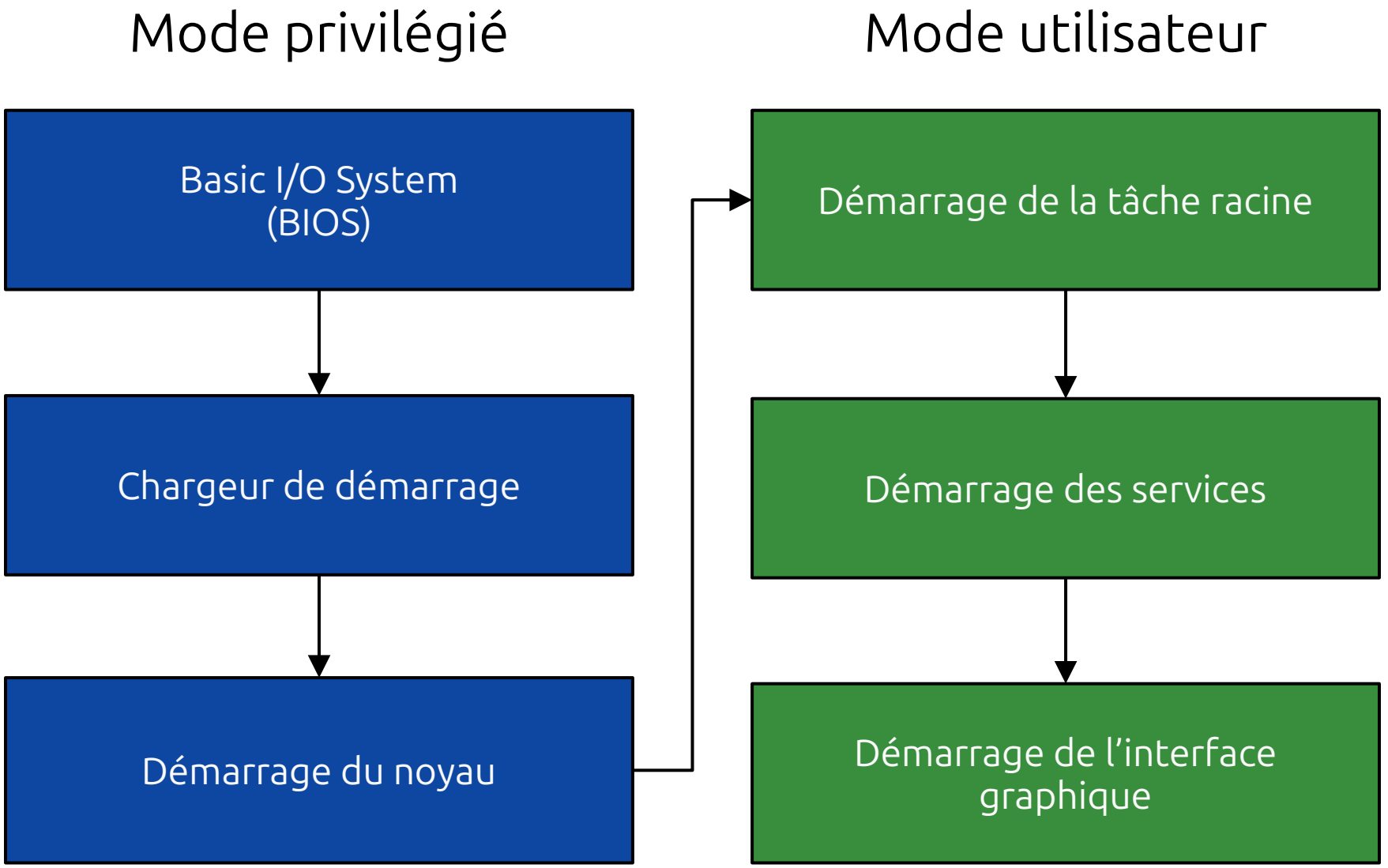
Conclusion: Les systèmes d’exploitations sont essentiels pour la fiabilité, la productivité et la protection des données pour une société connectée.

Survol du système d’exploitation

Modes d’exécution du processeur

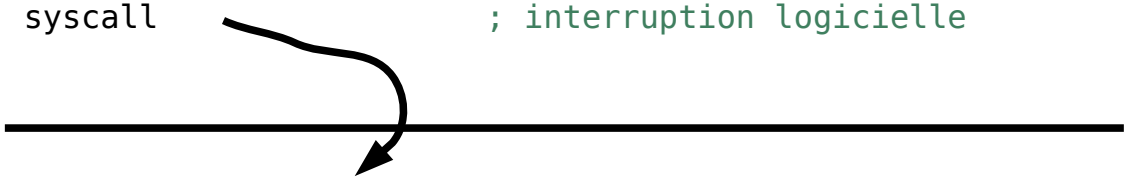
- Mode privilégié
  - Accès global à la machine
  - Sélectionne les tâches à exécuter
  - Peut interrompre ou terminer une tâche
  - Gère les interruptions matérielles
  - Attribue la mémoire
  - Arbitrage des accès aux périphériques
  - Libère les ressources utilisées en cas de crash d’une tâche
  - Espace dans lequel s’exécute le noyau du système d’exploitation
  - Crash du noyau est (souvent) fatal
- Mode utilisateur
  - Application en exécution
  - Limité à son propre espace mémoire
  - Accède aux ressources par des appels systèmes
  - Certaines instructions sont illégales
  - Le crash d’un programme est (généralement) bénin

Séquence de démarrage typique

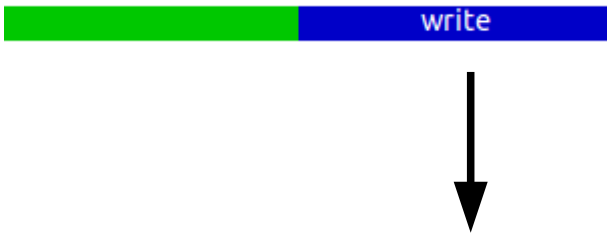


Appel système

```
; Ecrire msg sur la sortie standard
;
; prototype: int sys_write(unsigned int fd,
;                          const char *buf,
;                          size_t count)
;
mov     rdi, stdout     ; argument 1
mov     rsi, msg        ; argument 2
mov     rdx, msg_len    ; argument 3
mov     rax, sys_write  ; appel systeme dans rax
syscall                 ; interruption logicielle
```



Espace utilisateur (vert)



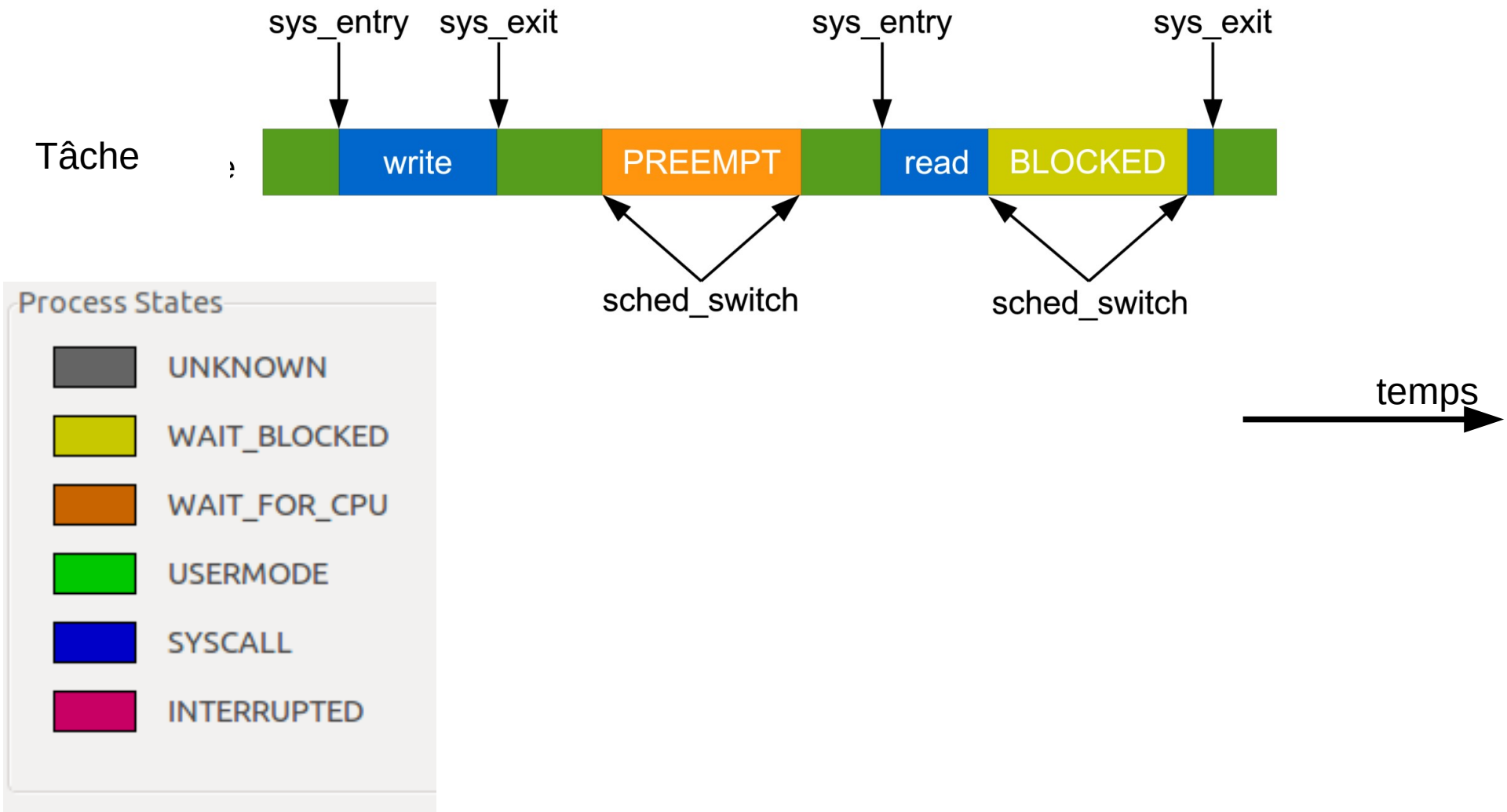
Espace noyau (bleu)

```
SYSCALL_DEFINE3(write, unsigned int, fd,
                const char __user *, buf,
                size_t, count)
{
     struct fd f = fdget_pos(fd);
     ssize_t ret = -EBADF;

     if (f.file) {
          loff_t pos = file_pos_read(f.file);
          ret = vfs_write(f.file, buf, count, &pos);
          if (ret >= 0)
               file_pos_write(f.file, pos);
          fdput_pos(f);
     }

     return ret;
}
```

Voir fichier read_write.c

Voir la liste complète dans:
arch/x86/entry/syscalls/syscall_64.tbl

Trace noyau


sys_entry
sys_exit
sys_entry
sys_exit
Tâche
write
PREEMPT
read
BLOCKED
sched_switch
sched_switch
temps
Process States
UNKNOWN
WAIT_BLOCKED
WAIT_FOR_CPU
USERMODE
SYSCALL
INTERRUPTED

linux/kernel/sched/core.c

```
/*
 * context_switch - switch to the new MM and the new
 * thread's register state.
 */
static inline void
context_switch(struct rq *rq, struct task_struct
*prev,
       struct task_struct *next)
{
    struct mm_struct *mm, *oldmm;

    prepare_task_switch(rq, prev, next);
    trace_sched_switch(prev, next);
    mm = next->mm;
    oldmm = prev->active_mm;
...
```

Tracepoint

Resulting trace event:

```
1649.424856148    kernel.sched_switch    { prev_pid = 24, next_pid =
1765 }
```

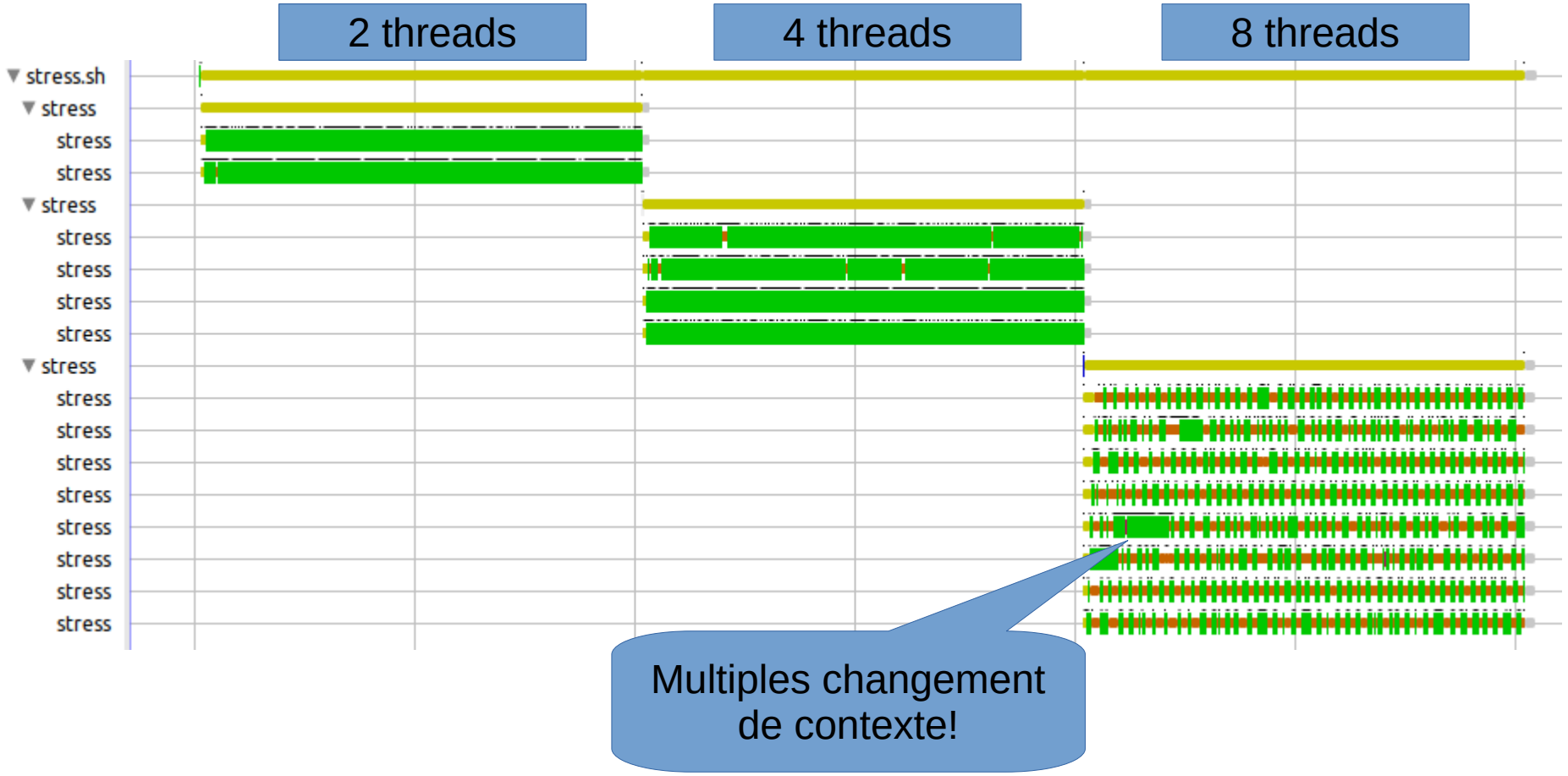
Timestamps (ns)

Event type

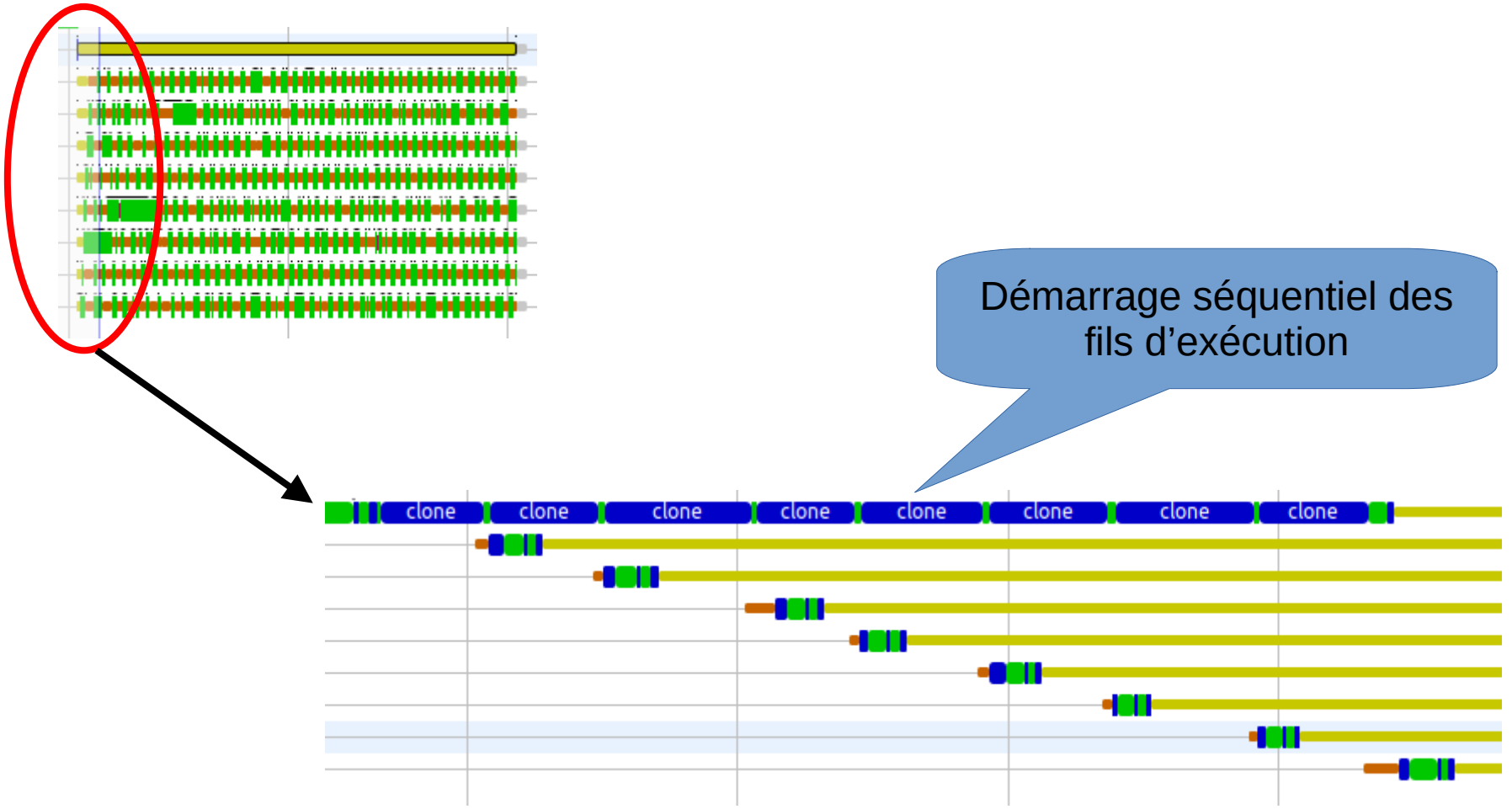
Payload

Ordonnancement

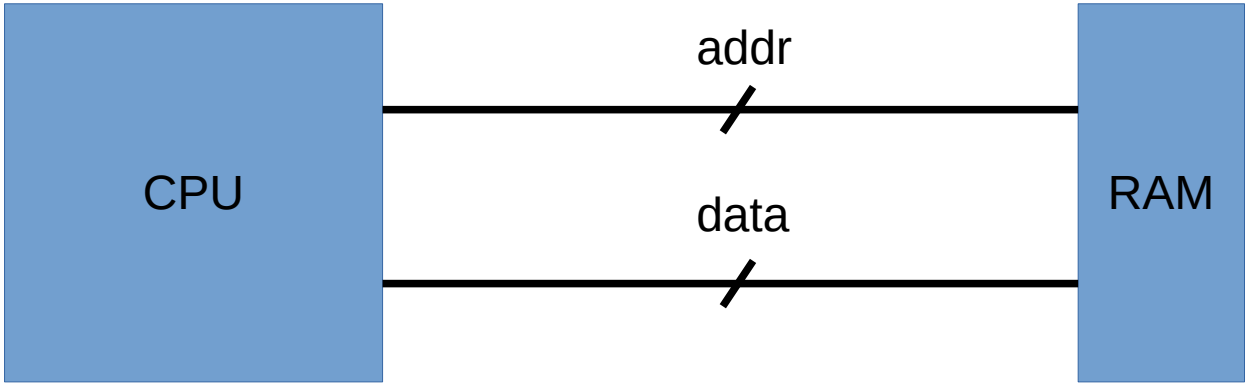
- Ordinateur à 4 processeurs



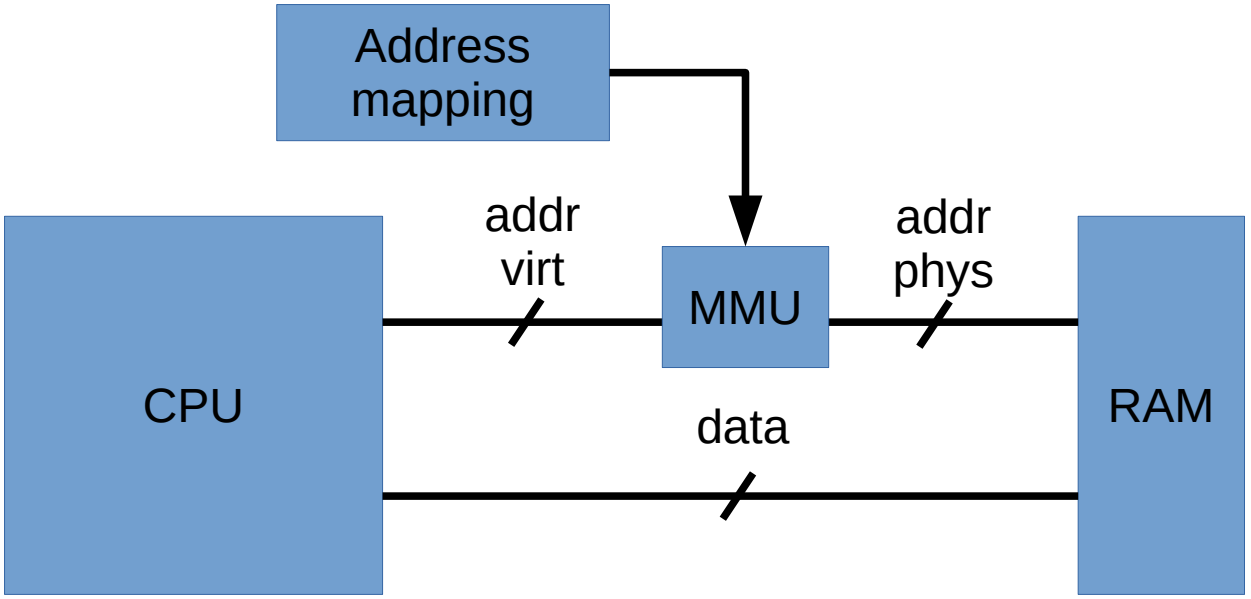
Démarrage d’un calcul parallèle



Adresses virtuelles



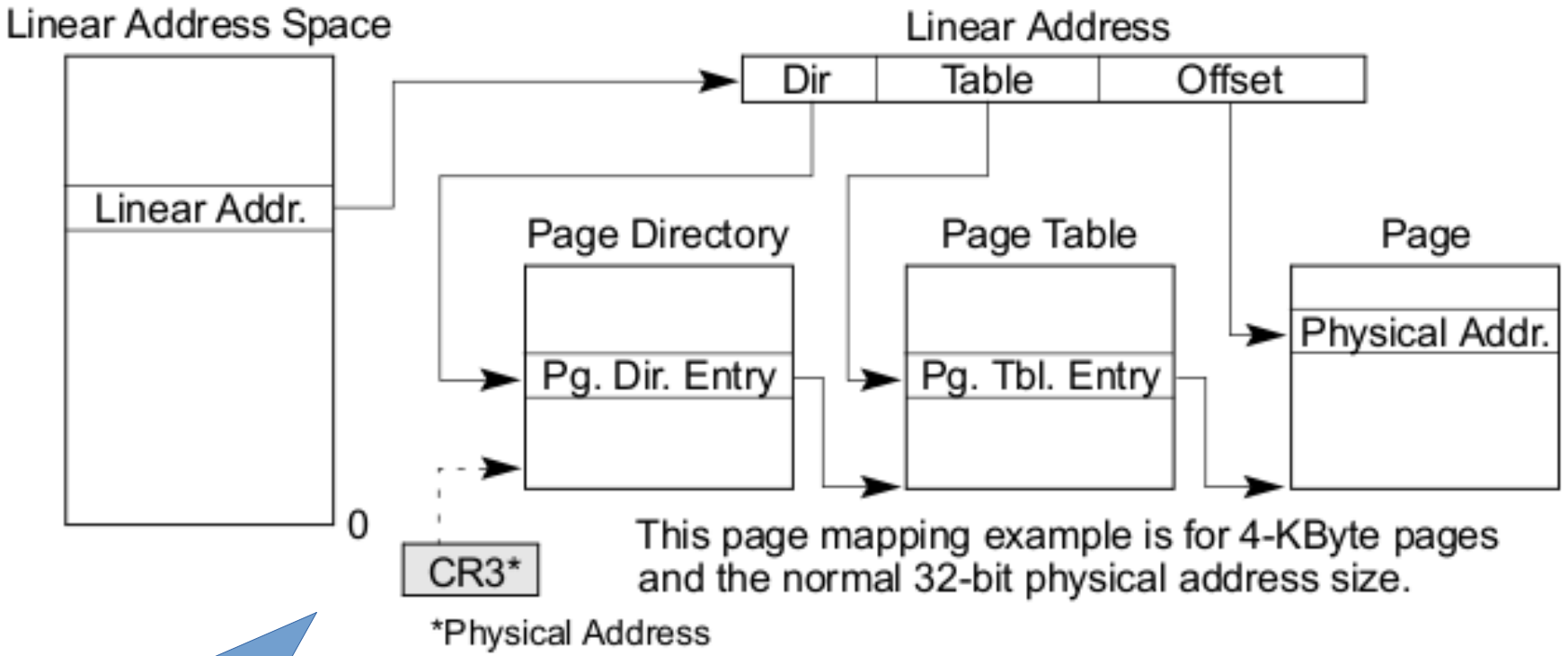
Adressage direct
(i.e. Arduino)



Adressage indirect

Memory
Management
Unit

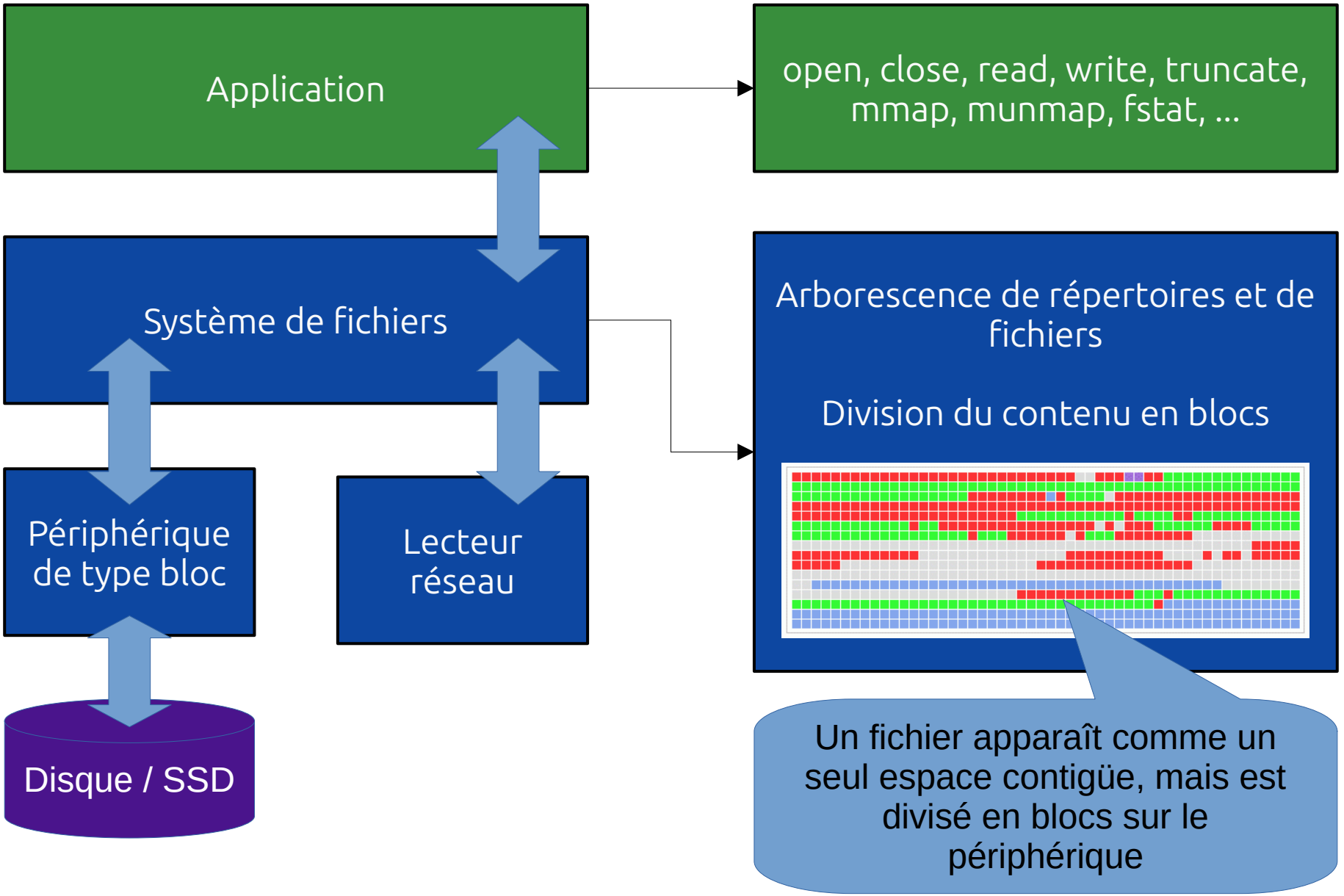
Table de page



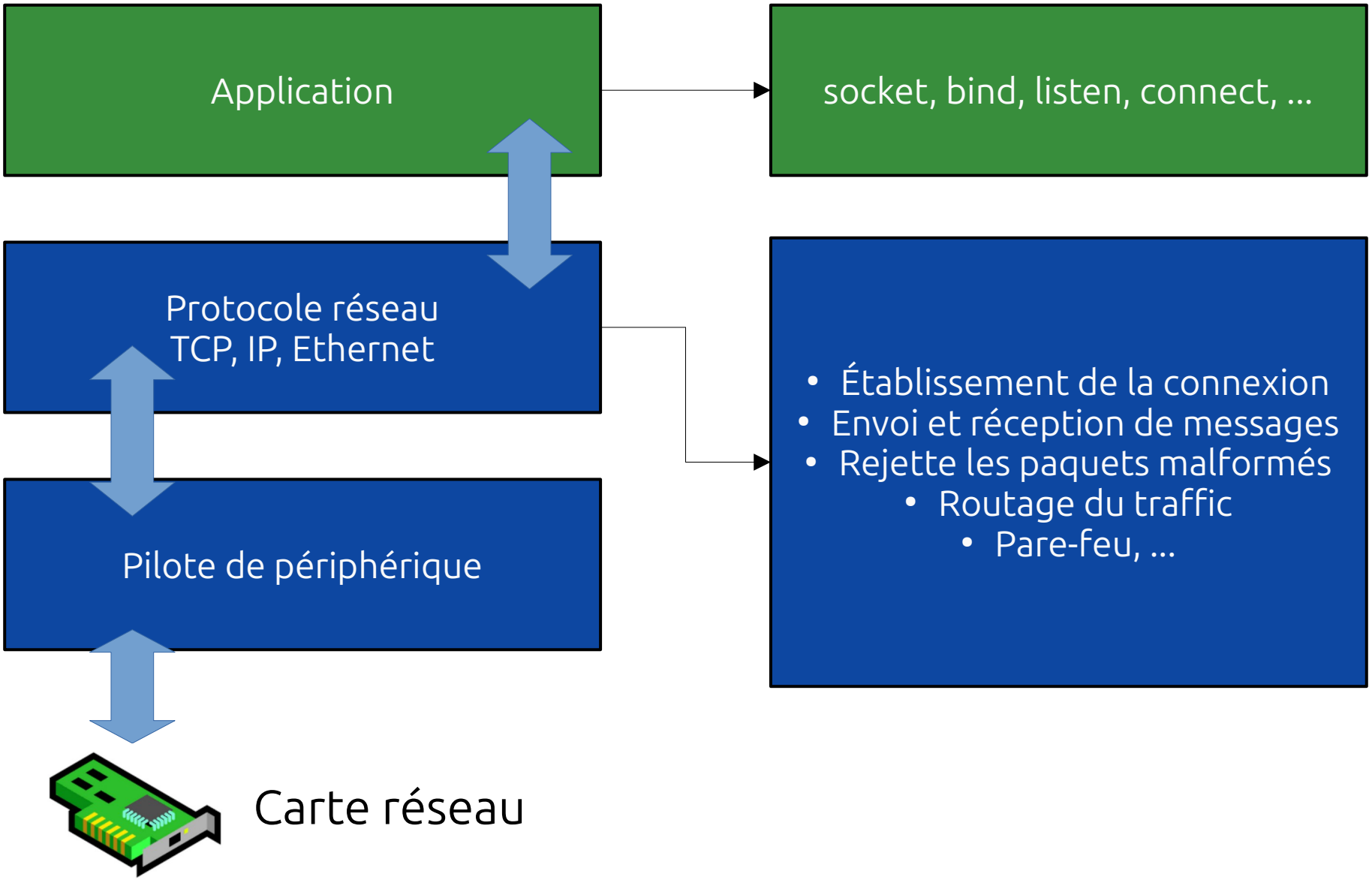
Le registre CR3 et la table de page sont accessibles en mode privilégié seulement

(Source : Intel® 64 and IA-32 Architectures Software Developer's Manua

Stockage permanent



Réseau

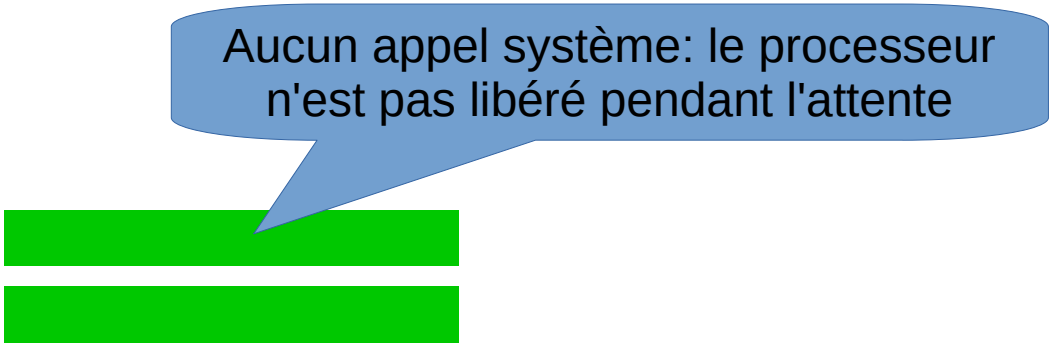

Application
socket, bind, listen, connect, ...
Protocole réseau
TCP, IP, Ethernet
• Établissement de la connexion
• Envoi et réception de messages
• Rejette les paquets malformés
• Routage du traffic
• Pare-feu, ...
Pilote de périphérique
Carte réseau

Minuterie

- Attente passive pour une durée déterminée
  - Essentiel dans les jeux, animation
  - Sécurité: limite de débit pour prévenir les dénis de services ratelimit
  - Exemples: nanosleep(), setitimer()
- Minuterie accessible seulement en mode privilégié
- Lorsque le délai est atteint, une interruption matérielle survient (traitement en mode privilégié)
- Des centaines/milliers d'échéances peuvent être demandées simultanément par différent programmes
- Solution: le système maintient une liste triée d'échéances et programme seulement la prochaine échéance

Synchronisation: attente active

- spinlock
  - Rapide: une instruction atomique (ex: cmpxchg)
  - Réservé pour une attente courte et probabilité de contention faible
  - Fréquent dans le noyau (ex: interruption)



Implémentation minimale

minispinlock.asm

```
1 ;
2 ; Implementation minimale d'un spinlock
3 ;
4 SECTION .data
5
6 SECTION .text
7 global mini_spin_lock
8 global mini_spin_unlock
9
10 mini_spin_lock:              ; adresse du verrou dans rdi
11     mov rcx, 1               ; valeur du verrou occupe
12 mini_spin_lock_retry:
13     xor rax, rax             ; remettre a zero rax
14     lock cmpxchg [rdi], rcx  ; compare la valeur [rdi] avec rcx
15                              ; si ([rdi] == rax) i.e. verrou occupe
16                              ;   ZF = 1, [rdi] = rcx
17                              ; sinon
18                              ;   ZF = 0, rax = [rdi]
19     jnz mini_spin_lock_retry ; si (ZF == 0), recommence
20     ret                      ; verrou obtenu
21
22 mini_spin_unlock:            ; adresse du verrou dans rdi
23     mov qword [rdi], 0       ; libere le verrou
24     ret
25
```

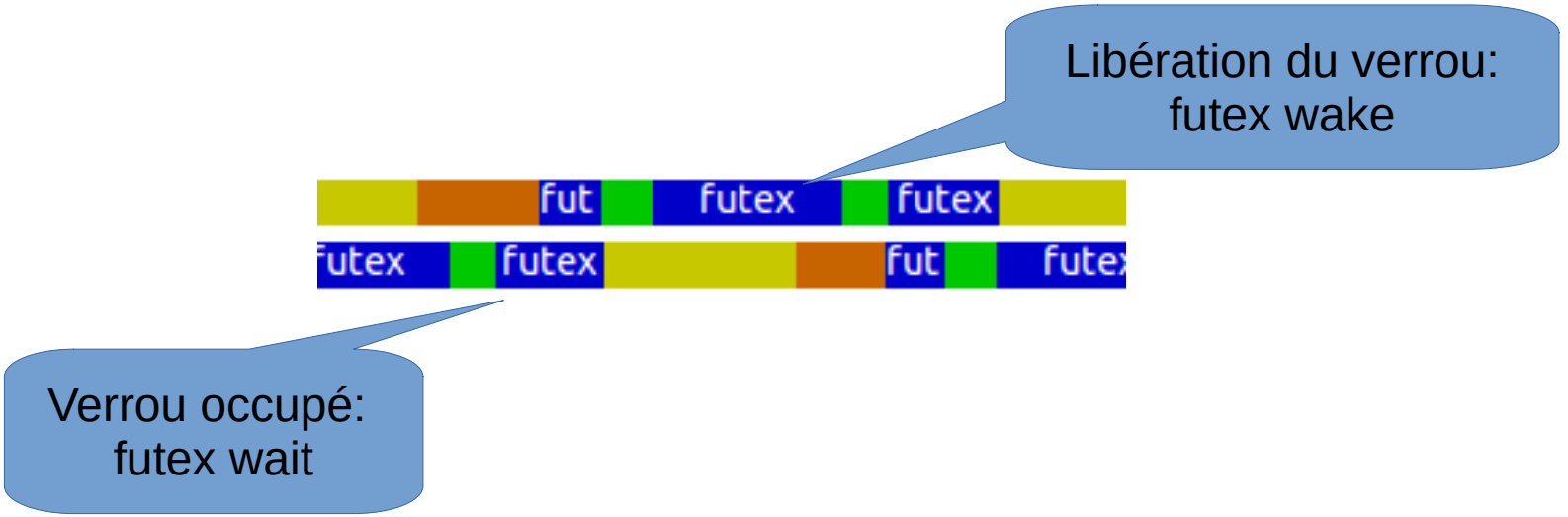
Assembleur

Interface

```
8  #ifndef MINISPINLOCK_H_
9  #define MINISPINLOCK_H_
10
11 void mini_spin_lock(int *lock);
12 void mini_spin_unlock(int *lock);
13
14 #endif /* MINISPINLOCK_H_ */
15
```

Synchronisation: attente passive

- Mutex implémenté avec futex()
  - Rapide si aucune contention: aucun appel système pour prendre le verrou (ex: cmpxchg)
  - Si le verrou est déjà utilisé:
    - futex(FUTEX_WAIT): attente passive (schedule)
    - futex(FUTEX_WAKE): verrou libéré



Programmes d’exemple

- Code source des exemples Arduino
- Traces
  - Télécharger Eclipse TraceCompass
- Scratchpad : répertoire s01
  - 01-hreset : instruction privilégiée
  - 02-syscall : appel système en assembleur
  - 03-page : scrutation de mémoire
  - 04-memwork : exemple espace mémoire
  - 05-crash : plantage de programme
  - 06-clang : rappels sur le langage C
  - 07-interop : mélanger le code C et C++
  - 08-visitor : patron du visiteur en C (pointeur de fonction)

À propos de CMake

- Générateur de makefile
- Permet de simplifier la compilation
- Pour ajouter du code:
  - Créer un répertoire
  - Ajouter add_subdirectory() dans fichier CMakeLists.txt à la racine
  - add_executable() pour compiler un exécutable
  - add_library() pour compiler une librairie
  - target_link_libraries() pour utiliser une librairie (entêtes + librairie)