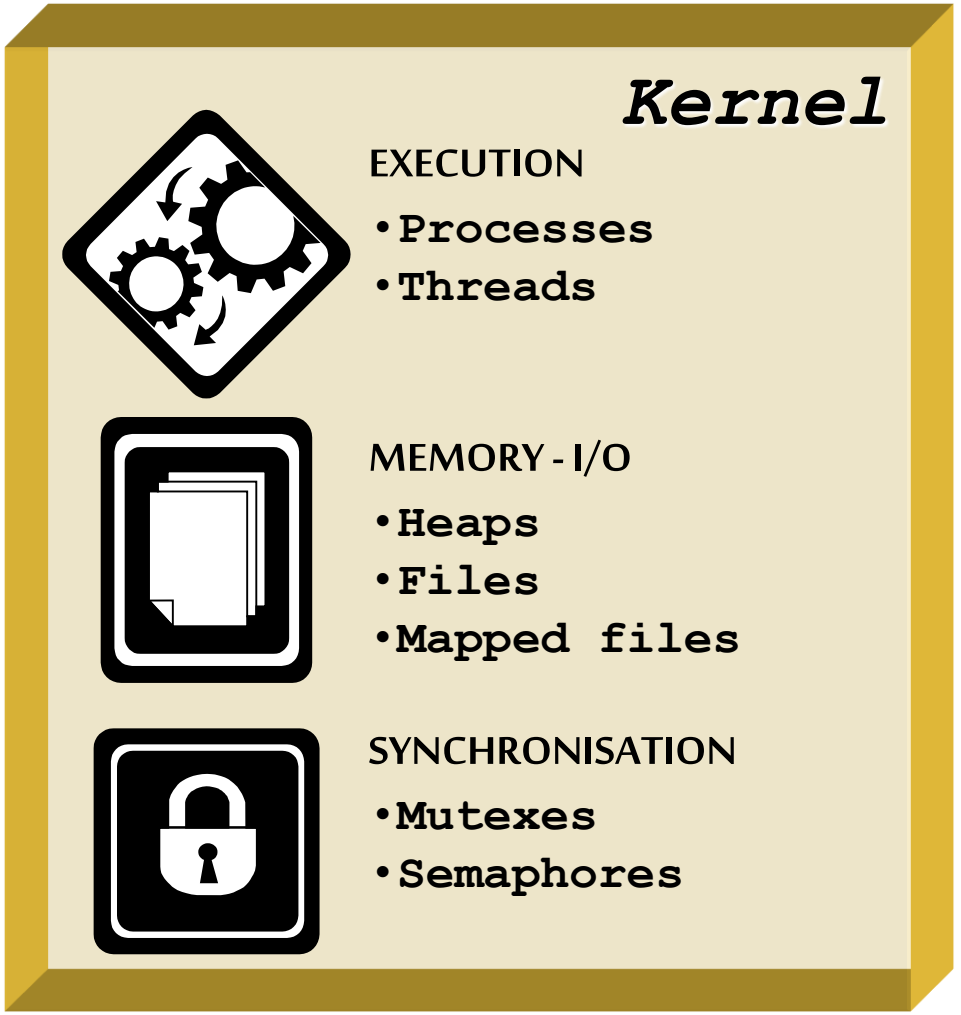
Développement d'applications natives

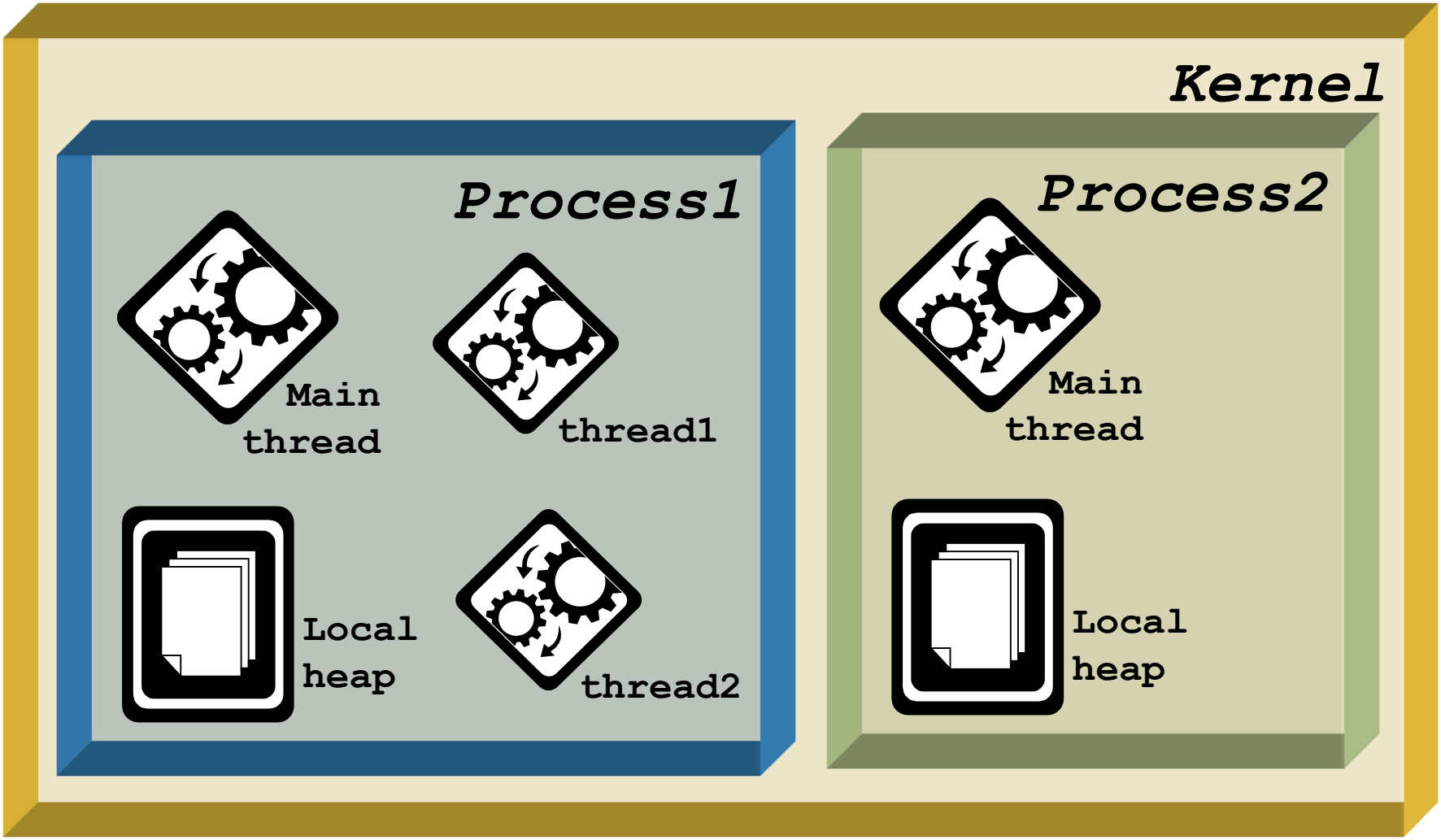
Processus & Thread UNIX

Objets noyau

- Services du noyau:
  - Exécution
  - Mémoire – I/O
  - Synchronisation
- Linux-API access
  - Par noms
  - Par type structure



Processus vs Thread



Processus

- Contexte mémoire
  - Un espace d’adressage virtuel
  - Un tas local (heap)
- Exécution
  - Image d’un ".out" (chargement du code)
  - Au moins un thread (main thread)
  - Créé lorsque l’exécutable est lancé

Processus – cycle de vie – I

- Création

  `pid_t fork()`

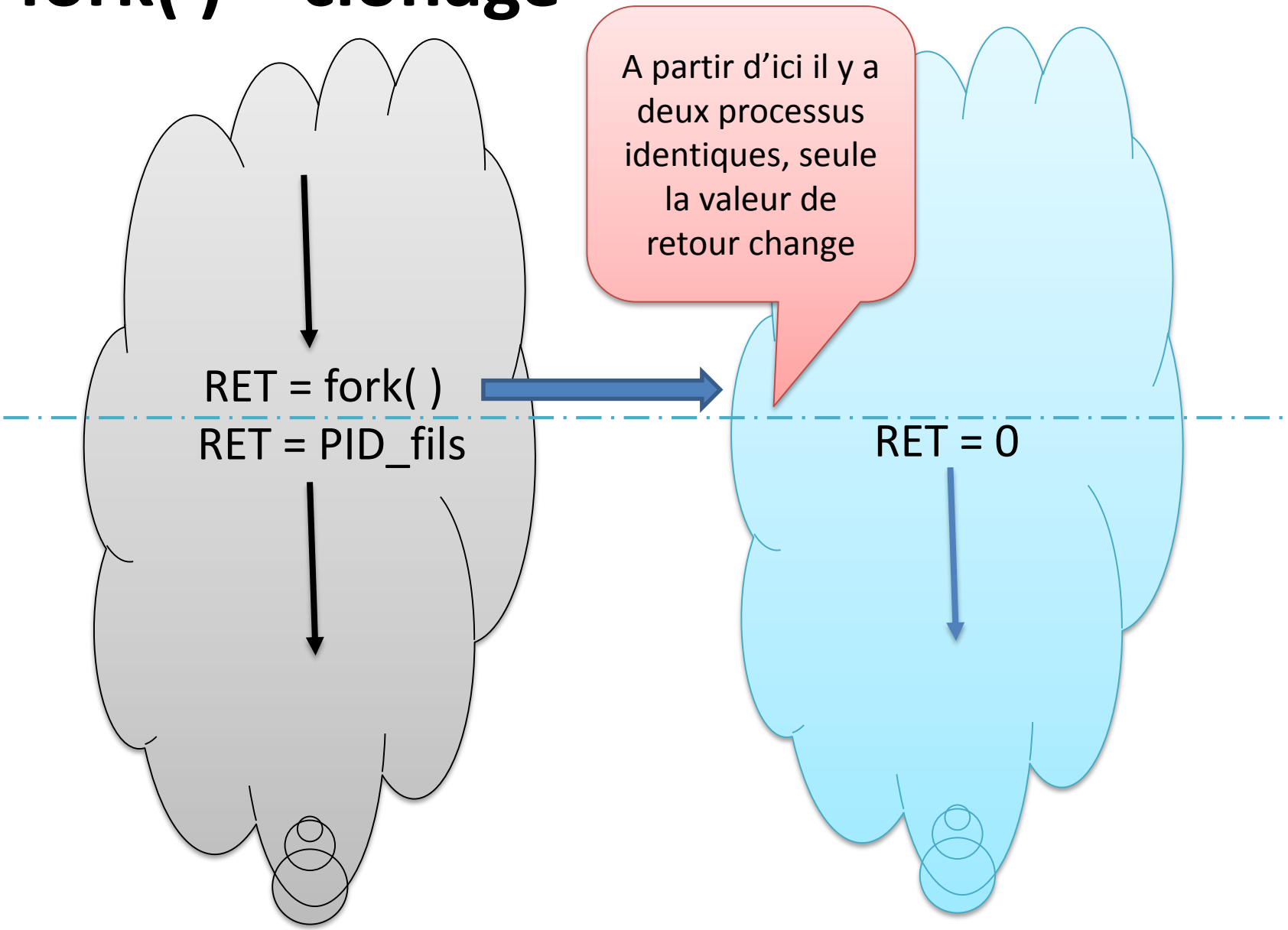
  Retourne: 0 → on est dans le fils
  >0 → on est dans le père
  -1 → echec

- Destruction
  - Depuis le processus lui-même (sortie)

    `void exit(int status);`
  - Depuis un autre processus

    `void kill(ProcessID, SIGNAL);`

fork( ) = clonage



Thread

- Contexte d'exécution préemptif
  - Pile, registres, exceptions, …
  - Priorité d'exécution

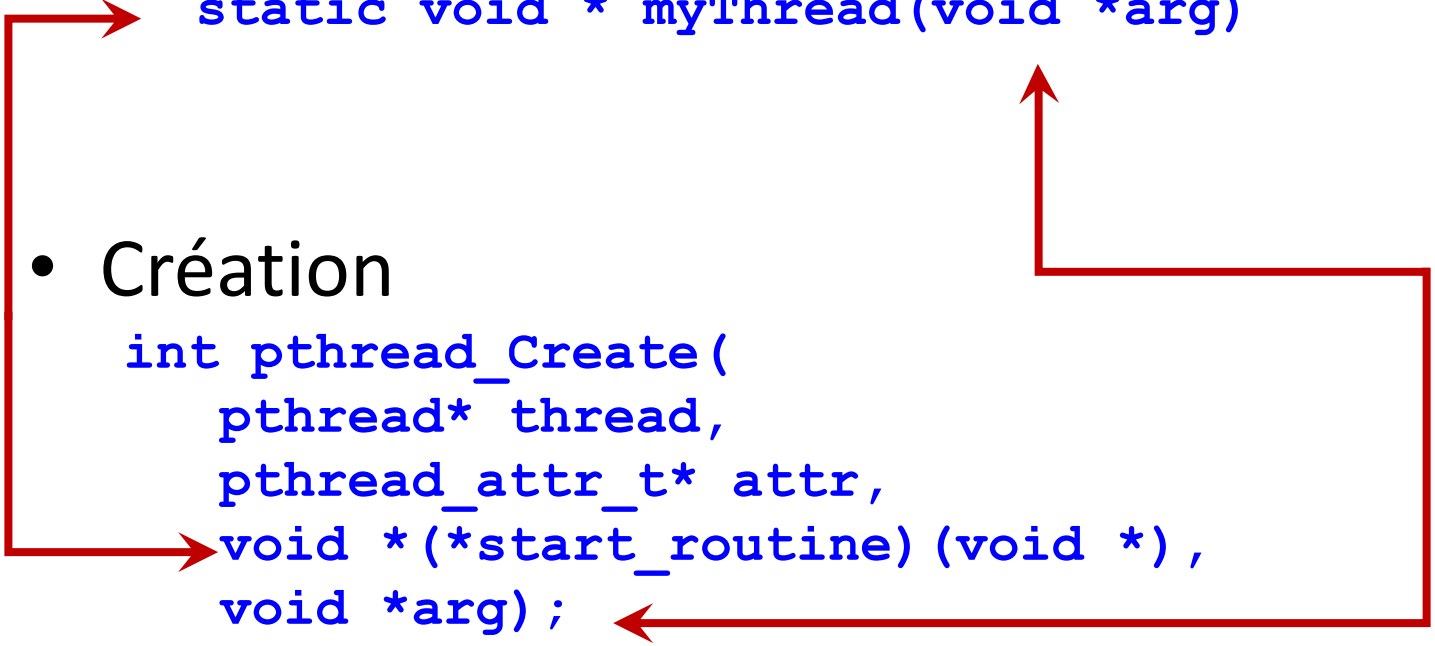
Thread – cycle de vie – I

- Point d'entrée

```
static void * myThread(void *arg)
```

- Création

```
int pthread_Create(
    pthread* thread,
    pthread_attr_t* attr,
    void *(*start_routine)(void *),
    void *arg);
```



Thread – cycle de vie – II

- Destruction

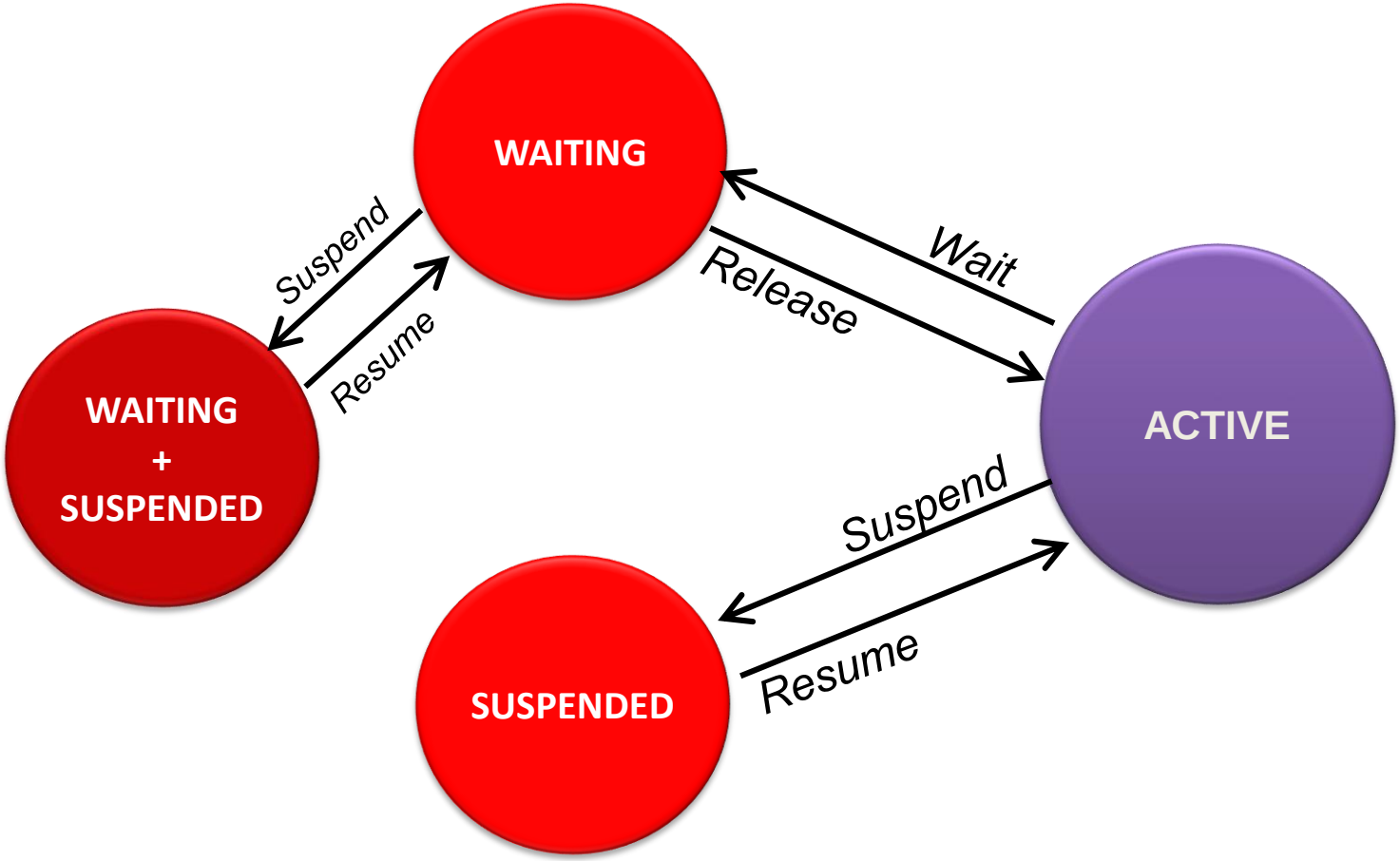
```
void pthread_exit (void * retval);

int pthread_cancel (pthread_t thread);

int pthread_setcancelstate (int state,
                            int * etat_pred);
```

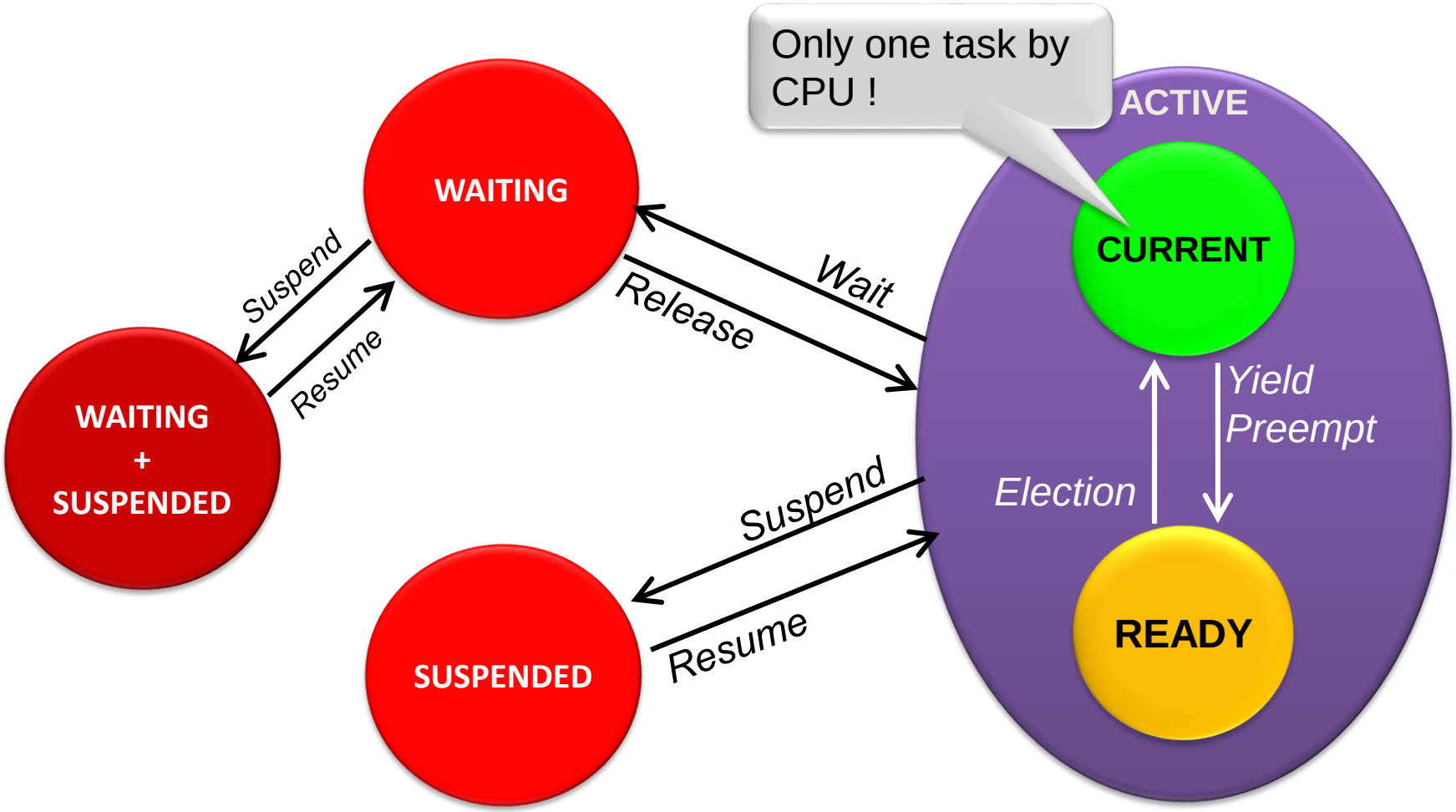
Etats d'un thread

Point de vue du développeur



Etats d'un thread

Point de vue de l'ordonnanceur



Priorités d'un thread et police d'ordonnancement I

Plage de priorité en fonction de la politique d'ordonnancement:

- `int sched_get_priority_max(int policy);`
- `int sched_get_priority_min(int policy);`

*Policy = {*SCHED_FIFO, SCHED_RR (Round-robin), SCHED_DEADLINE, SCHED_OTHER, SCHED_BATCH, SCHED_IDLE*}*

Priorités d'un thread et police d'ordonnancement II

```
int pthread_setschedprio(pthread_t thread, int
prio);

int pthread_setschedparam(pthread_t thread, int
policy, const struct sched_param *param);

int pthread_getschedparam(pthread_t thread, int
*policy,struct  sched_param *param);

struct sched_param {
        int sched_priority; /* Scheduling priority*/
      };
```