



Systèmes Embarqués 1 & 2

Classes T-2/I-2 // 2018-2019

p.02 – Systèmes d'exploitation

Exercices

Exercice 1

Concevoir et réaliser un micro noyau coopératif en C pour un μ P ARM. Seules les routines accédant des ressources du processeur non disponible en C devront être réalisées en assembleur.

L'interface pour les programmes applicatifs devra inclure les services suivants :

- (a) Méthode pour initialiser le micro noyau (**kernel_init**)
- (b) Méthode pour lancer/activer le micro noyau (**kernel_launch**)
- (c) Méthode pour ajouter un nouveau tcb au micro noyau (**kernel_add_tcb**)
- (d) Méthode pour initier la commutation de contexte (**kernel_yield**)

Exercice 2

Concevoir et réaliser pour le micro noyau des exercices précédants, un jeu de routines permettant la gestion de threads.

L'interface pour les programmes applicatifs devra inclure les services suivants :

- (a) Méthode pour créer un thread (**thread_create**)
- (b) Méthode pour terminer un thread (**thread_exit**)
- (c) Méthode pour initier la commutation de contexte (**thread_yield**)

Le micro noyau devra gérer dynamiquement les threads. Il permettra de choisir la taille de leur pile (stack), mais il garantira une profondeur minimale de 1000 mots de 32 bits. Les threads auront le prototype suivant :

```
void thread (void* param);
```

Exercice 3

Concevoir et réaliser pour le micro noyau des exercices précédants, un jeu de routines permettant l'opération de sémaphores.

L'interface pour les programmes applicatifs devra inclure les services suivants :

- (a) Méthode pour créer un sémaphore (**sema_create**)
- (b) Méthode pour détruire un sémaphore (**sema_destroy**)
- (c) Méthode pour signaler un sémaphore (**sema_signal**)
- (d) Méthode pour attendre sur un sémaphore (**sema_wait**)

**Exercice 4**

Concevoir et réaliser pour le micro noyau des exercices précédants, un jeu de routines permettant l'opération de files d'attente de message (*message queue*).

L'interface pour les programmes applicatifs devra inclure les services suivants :

- (a) Méthode pour créer une "message queue" (**msgq_create**)
- (b) Méthode pour détruire une "message queue" (**msgq_destroy**)
- (c) Méthode pour poster un message (**msgq_post**)
- (d) Méthode pour prendre un message (**msgq_fetch**)

Exercice 5

Etendre le jeu de méthodes avec un service pour suspendre l'exécution d'un thread pour une certaine période de temps (**thread_delay**).