TP n°2 : Sleep et Entrées / sorties

1 Comment rendre ces TP ?

Les TPs 1 et 2 doivent être rendus ensemble sur AMETICE [1]
au plus tard le mardi 21 novembre 2023 à 18h30.

Vous devez rendre ces TP (seul(e) ou avec une autre personne) sous la forme d'une archive ZIP sur AMETICE [2] dont le nom est composé à partir de votre numéro de groupe suivi de vos noms de famille (en majuscules) et vos prénoms (en minuscules). Par exemple

```
G1_NOM1_prenom1_NOM2_prenom2.zip
```

2 Ajouter un thread idle

Nous avons un ordonnanceur et nous avons déjà exécuté deux threads en simultané. Nous allons en ajouter un troisième (qui boucle sans rien faire) afin de s'assurer que nous aurons toujours **au moins un thread prêt** à choisir et à exécuter.

Travail à faire :

- Prévoir une variable globale :

```
PSW idle;
```

- Prévoir (lors de l'initialisation du système) la création d'un code simple qui effectue une boucle infinie :

Un code qui boucle dans system_init

```
...
idle.PC = 120;
assemble_string(idle.PC, "loop: jump loop");
...
```

- Modifier l'ordonnanceur afin qu'il renvoie `idle` si aucun processus n'est prêt (ce qui va arriver dans les questions qui suivent).

3 Endormir des threads

On se propose de réaliser l'appel système `sysc SYSC_SLEEP` qui va endormir le thread courant pendant `AC` seconde(s).

Travail à faire :

- Ajoutez un état endormi.
- Ajoutez une date de réveil (voir `man 2 time`) dans le tableau des threads.
- Endormir le thread courant (voir exemple ci-dessous).

1. ref:ametice
2. ref:ametice

Exemple d'endormissement d'un thread

```
define SYSC_EXIT          100
define SYSC_PUTI          200
define SYSC_NEW_THREAD    300
define SYSC_SLEEP         400

set 4                  // AC = 4
sysc SYSC_SLEEP        // endormir AC sec.
sysc SYSC_PUTI         // afficher AC
sysc SYSC_SLEEP        // endormir AC sec.
sysc SYSC_PUTI         // afficher AC
sysc SYSC_EXIT         // fin du thread
```

- Faites en sorte de réveiller les endormis (complétez la fonction `wakeup` et ajoutez un appel à chaque tour complet de l'ordonnanceur).

4 La fonction getchar

On se propose de réaliser l'appel système `SYSC SYSC_GETCHAR` qui va lire un caractère sur l'entrée standard et le placer dans `AC` ou attendre l'arrivée d'un caractère.

Nous ne pouvons pas réellement utiliser le clavier car cela impose de contrôler parfaitement les arrivées de caractères. La couche matérielle (`cpu.c`) simule l'arrivée d'un caractère toutes les trois secondes et signale cet événement par une interruption clavier. Le caractère en question se trouve dans le registre `IO` . (voir `cpu.h`).

Travail à faire :

- Prévoir la définition du tampon (capacité un caractère) :

```
char tampon = '\0';     /* le '\0' indique le vide */
```

- Ajouter un nouvel état `GETCHAR` (endormi en attente de caractère).
- Endormir le thread courant sur une demande de caractère si le tampon est vide. Si le tampon n'est pas vide, renvoyer le caractère.
- Prévoir une fonction système `keyboard_event()` appelée par le système sur interruption clavier. Cette fonction va
  - ▷ chercher un thread dans un état `GETCHAR` , lui donner le caractère arrivé et le réveiller,
  - ▷ ou stocker le caractère dans le tampon (si aucun thread n'est trouvé).
- Vous pouvez tester cette fonction en créant un thread qui tente de lire un caractère toutes les secondes ou toutes les quatres secondes :

Lecture d'un caractère et endormissement

```
        define SYSC_EXIT        100
        define SYSC_PUTI        200
        define SYSC_NEW_THREAD  300
        define SYSC_SLEEP       400
        define SYSC_GETCHAR     500

loop:   sysc SYSC_GETCHAR     // AC = getchar()
        sysc SYSC_PUTI        // puti(AC)
        set 1                 // AC = 1
        sysc SYSC_SLEEP       // sleep(AC)
        jump loop
```

Amélioration :

- Quand il n'y a plus de caractère, le thread reste éternellement bloqué.
- Modifier le système afin de tuer le thread (au bout de dix secondes) quand il n'y a plus de caractère (il faut enrichir la fonction `wakeup`).