# Programmation Système
# TP noté (1h30)

## Transmission en morse

Cet exercice a pour but de réaliser la transmission d'un message entre deux processus au moyen de signaux en utilisant le code morse. Un message en morse est représenté par une chaîne de caractères dont chaque caractère est soit `'.'` (point), soit `'_'` (tiret), soit `' '` (espace). On procédera de la manière suivante :

1. Écrire un programme `receiver1.c` qui commence par afficher son pid, puis boucle indéfiniment et affiche le caractère :
   – `'.'` dès qu'il reçoit le signal `SIGUSR1` ;
   – `'_'` dès qu'il reçoit le signal `SIGUSR2` ;
   – `' '` dès qu'il reçoit le signal `SIGINT`.
   (On utilisera l'instruction `fflush(stdout);` pour que le caractère s'affiche immédiatement sans retour à la ligne.) Le programme doit terminer automatiquement s'il ne reçoit aucun de ces 3 signaux en l'espace de 10 secondes.

   Exemple d'exécution (en utilisant deux fenêtres terminal) :

   ```
   $ ./receiver1                    $ kill -USR1 365
   Je suis 365                      $ kill -INT 365
   '. _'                            $ kill -USR2 365
   $                                $
   ```

2. Écrire un programme `sender1.c` qui prend en argument (sur la ligne de commande) un message en morse (par exemple : `'. _._. .. _._.'`) et le pid d'un processus, et envoie les signaux correspondants au processus. Essayez ce programme avec le programme `receiver1.c` : lancer d'abord `receiver1` dans une fenêtre, puis `sender1` dans une autre. Est-ce que le programme `receiver1.c` affiche le bon message ? Pourquoi ? Mettre la réponse à ces questions en commentaire dans le fichier `sender1.c`.
3. Écrire deux programmes `sender2.c` et `receiver2.c` qui s'échangent leur pid en utilisant un segment de mémoire partagée. Le programme `sender2.c` est chargé de créer et de détruire le segment (dont la taille doit être assez grande pour contenir deux pids). Chaque programme affiche son pid, l'écrit en mémoire partagée, attend que l'autre programme ait fait de même, et enfin affiche le pid de l'autre programme avant de terminer.

Exemple d'exécution :

```
$ ./sender2                          $ ./receiver2
Je suis 365                          Je suis 366
L'autre est 366                      L'autre est 365
$                                    $
```

4. Écrire deux programmes `sender3.c` et `receiver3.c` qui s'échangent leur pid comme dans la question 3, réalise la transmission d'un message en morse en utilisant les signaux comme dans la question 2, mais en plus se synchronisent en utilisant le signal `SIGTERM` pour que le message soit transmis de manière fiable.

   Exemple d'exécution :

```
$ ./sender3 '._ _... _._.'          $ ./receiver3
J'envoie '._ _... _._.'             J'ai recu '._ _... _._.'
$                                    $
```