2ème année 2014-2015

Synchronisation entre processus

Février/Mars 2015

Objectifs :
— utiliser les appels systèmes liés aux processus Unix ;
— comprendre le mécanisme des signaux.

1 Masquage de signaux

L'appel système suivant permet de positionner le masque avec la valeur passée en paramètre et de mettre le processus en attente d'un signal (conformément au masque).

```
#include <signal.h>
int sigsuspend(const sigset_t *set);
```

Quant à l'appel suivant

```
#include <signal.h>
int sigpending(sigset_t *set);
```

il permet de déterminer les signaux "en attente", c'est-à-dire dont au moins une occurence n'a pas été délivrée (pour cause de masquage).

▷ **Exercice 1 : Désactiver le compteur**

Modifier l'exercice 3 de la séance précédente afin de permettre à l'utilisateur d'invalider le comptage des événements lorsqu'il appuie sur `Ctrl-C`*, puis de le réactiver de la même façon.*

On fera en sorte que les événements arrivés pendant une période de masquage ne soient pas pris en compte. ■

2 Exécution d'une commande

La fonction suivante permet d'exécuter une commande Unix

```
#include <unistd.h>
int execvp(const char *file, char *const argv[]);
```

Le premier argument est le nom du programme à exécuter, le second est un tableau contenant les différents paramètres qui seront passés à la commande (et accessible par elle en temps que `argv[0]`, `argv[1]` ... où `argv[0]` est le nom du programme et prend généralement la même valeur que `file`).

Cette fonction n'est pas censée avoir de valeur de retour, puisque si l'exécution du nouveau programme est possible, le programme initial (celui qui contient l'appel à `execvp`) est entièrement remplacé par ce nouveau programme.

L'appel système suivant arme une alarme pour un nombre de secondes choisi (mais aucune attente n'est effectuée par cet appel).

```
#include <unistd.h>
unsigned int alarm(unsigned int seconds);
```

Lorsque le délai est écoulé, le signal `SIGALRM` est délivré au processus (s'il n'est pas masqué, naturellement).

▷ **Exercice 2 : Limitation du temps d'exécution**

Écrire la commande `timeout <temps> <commande> [<arg1> ...]` *qui exécute dans un processus fils la commande dont le nom est passé en paramètre avec les éventuels arguments. Le processus père attend alors la fin du processus exécutant la commande ou tue ce processus si il ne s'est pas terminé avant le temps passé en paramètre.* ■