TP - Les signaux

Préambule

Recopiez le répertoire *./sources*
`kill -l` (pour avoir la liste des signaux disponibles)
/usr/include/sys/signal.h, /usr/include/signal.h, /usr/include/linux/signal.h, etc.
(pour la programmation des signaux)

Notes de cours : les principes et les primitives.

1 Prise en compte

a) Regardez `prg1.c` et testez-le avec le caractère de controle `intr` (*`Ctrl-C`*) qui provoque l'envoi du signal *`SIGINT`*.

Expliquez l'absence d'écho sur le terminal.

b) Ajoutez-y un traitant pour le signal `SIGTSTP (Ctrl-Z)` qui affiche

```
---- suspension du processus ----
```

et suspend effectivement le processus comme le fait le traitant par défaut sur la frappe de *`Ctrl-Z`*.

c) Regardez `prg2.c` et exécutez-le en tapant :

```
abcdef<RC>ghi<Ctrl-C>klm<Ctrl-C>nop
```

de façon à mettre en évidence l'échec du `scanf` (qui fait un `read` sur le terminal) à la réception d'un signal.

Est-il important que le signal soit `SIGINT` ?

2 Masquage

Exécutez `prg3` en l'interrompant par un `Ctrl-C`, puis enlevez le commentaire devant un des appels à `sigprocmask` et ré-exécutez le pour chacun des deux cas.

Expliquez la différence de comportement sur l'apparition du message "*`fin du processus fils`*".

3 Attente

a) Exécutez plusieurs fois `prg4` et expliquez la différence de nombre de signaux envoyés et reçus.

b) Afin d'éviter la perte de signaux on instaure un protocole de communication. Le père envoie un accusé de réception lorsqu'il a reçu le signal et le fils se bloque en attente (`pause`).

Expliquez les raisons de l'interblocage sur *`prg4bis`*.

Pour aller plus loin :

c) Evitez le problème prééceédent en vous bloquant avec `sigsuspend`.

d) Point de reprise : regardez `prg5.c` et exécutez-le.

4 Applications

a) **Endormissement d'un processus** : en utilisant `SIGALRM`, écrivez la fonction `sleep` de la bibliothèque C.

b) **Terminaison processus fils** : il peut arriver que le processus père ait des traitements à effectuer avant de se bloquer sur un `wait`. Ecrivez un traitant pour le signal `SIGCHLD` émis automatiquement par le système à la terminaison d'un processus fils pour pouvoir lire le code de retour d'un processus fils sans avoir à se bloquer dans le père.