Signaux

François Trahay

Signaux

Rappel (CSC3102)

- Signal: mécanisme de communication inter-processus
- Message: un entier entre 1 et 31
- Ordre de réception aléatoire (différent de l'ordre d'émission)
- Une routine de réception est automatiquement invoquée chez le récepteur dès que le signal arrive

Envoyer un signal

- `int kill(pid_t pid, int sig);`
  - Envoie le signal `sig` au processus `pid`
  - Quelle valeur pour `sig`?
    - valeur entière (par ex: `9`): pas portable (dépend de l'OS)
    - constante (par ex: `SIGKILL`) définie dans `signal.h`

Recevoir un signal

- `int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);`
  - Spécifie le comportant lors de la réception du signal `signum`
  - `struct sigaction` est une structure de la forme:

```c
struct sigaction {
        void     (*sa_handler)(int); // pointeur sur la fonction à appeler
        void     (*sa_sigaction)(int, siginfo_t *, void *);
        sigset_t   sa_mask;
        int        sa_flags;
        void     (*sa_restorer)(void);
};
```

Attendre un signal

- `int pause( );`
  - Attend qu'un signal (non ignoré) soit reçu

Programmer une alarme

- `int alarm(unsigned int s);`
  - Programme l'envoi de `SIGALRM` après `s` secondes

Pour aller plus loin

Les 3 sous-sections suivantes présentent des notions pour les étudiant·e·s Ninja qui ont vraiment envie d'aller encore plus loin.

`sigsetjmp` et `siglongjmp`

Permet de faire un “saut (`goto`) non local”

- `int sigsetjmp(sigjmp_buf env, int savesigs);`
  - sauvegarde l’environnement d’appel courant (pile d’appel, pointeur d’instruction, etc.)
- `void siglongjmp(sigjmp_buf env, int val);`
  - restaure l’environemment sauvegardé `env`
  - le programme continue son exécution comme s’il retournait de la fonction `sigsetjmp`

Programmation événementielle

libuv permet de faire de la programmation événementielle

- “Asynchronous I/O made simple” dit le site de `libuv`.
- “Networking in `libuv` is not much different from directly using the BSD socket interface, some things are easier, all are non-blocking, but the concepts stay the same. In addition libuv offers utility functions to abstract the annoying, repetitive and low-level tasks like setting up sockets using the BSD socket structures, DNS lookup, and tweaking various socket parameters.” (extrait du chapitre Networking de la documentation libuv).

Coroutines

- inconvénient de la programmation événementielle: définition de nombreuses functions de *callback*
  - callback appelé quand un appel asynchrone est terminé
  - réduit la lisibilité du code
- solution: **coroutines**
  - possibilité de suspendre l'exécution d'une fonction pour la reprendre plus tard