Programmation système

TP 7 – Gestion des signaux

Exercice 1: Redimensionnement du terminal

De nombreuses applications en mode texte (e.g., `vim`, `nano`, `htop`, `ranger`, …) adaptent leur affichage à la taille de la fenêtre du terminal. Lorsque la fenêtre est redimensionnée par l'utilisateur, son contenu est automatiquement redessiné. Dans cet exercice, vous allez voir comment cela fonctionne.

1. Écrivez un programme qui affiche les dimensions de son terminal de contrôle. Par exemple, si le terminal a une largeur de 80 caractères et une hauteur de 24 lignes, le programme doit afficher :

   ```
   Window size: 80 x 24
   ```

   Vous pouvez obtenir ces informations avec le code suivant (nécessitant `<sys/ioctl.h>`) :

   ```
   struct winsize ws;
   try(ioctl(STDIN_FILENO, TIOCGWINSZ, &ws));  // Fails if stdin is not a terminal
   ```

   Après cet appel, les dimensions du terminal sont stockées dans `ws.ws_col` et `ws.ws_row`.

2. Modifiez votre programme pour qu'il affiche les nouvelles dimensions à chaque fois que l'utilisateur change la taille de la fenêtre. Par exemple :

   ```
   Window size: 80 x 24        Après avoir lancé le programme
   Window size: 114 x 31       Après le 1er redimensionnement
   Window size: 60 x 42        Après le 2ème redimensionnement
   ...
   ```

   Pour détecter les redimensionnements, vous devez installer un gestionnaire (`en: handler`) pour le signal `SIGWINCH`. C'est ce signal qui est généré par le noyau lorsque la taille du terminal change. Servez-vous, dans un premier temps, de l'appel système `signal()`.

3. Réécrivez le programme précédent en remplaçant `signal()` par l'appel système `sigaction()`. Pensez à bien initialiser la structure `sigaction` que vous passez en deuxième paramètre.

4. *Bonus :* Si vous avez envie, vous pouvez rendre votre programme plus visuel en remplaçant le message de la forme "`Window size: 80 x 24`" par un rectangle ayant exactement les dimensions du terminal. Les caractères de dessin de boîte [1] sont très utiles pour cela : ┌ ┐ └ ┘ ─ │

Exercice 2: Mesure de vitesse

1. Écrivez un programme `speed` qui affiche une fois par seconde la vitesse à laquelle il reçoit des données sur son entrée standard (en octets par seconde). Exemple d'utilisation :

   ```
   $ while true; do echo "xxx"; sleep 0.001; done | ./speed
   1336 B/s
   1272 B/s
   1228 B/s
   ```

1. https://en.wikipedia.org/wiki/Box-drawing_character

```
1224 B/s
1412 B/s
...
```

(Dans cet exemple, les données proviennent de la boucle `while` exécutée par le shell, et elles sont envoyées au programme `speed` à travers un tube.)

Pour faire cet exercice, servez-vous de l'appel système `alarm()`, qui demande au noyau de générer le signal `SIGALRM` après le nombre de secondes donné en argument. Dans un premier temps, faites l'affichage de vitesse depuis le gestionnaire que vous installerez pour ce signal.

2. En général, un gestionnaire de signal ne devrait pas appeler de fonctions de la bibliothèque `stdio`, telles que `printf()`, car ces fonctions ne sont pas réentrantes. Réécrivez le programme précédent de manière à ce que l'affichage soit fait dans le programme principal, et que le gestionnaire de signal soit le plus simple possible.

Exercice 3: Limitation de vitesse

Écrivez un programme `slowcat` qui copie son entrée standard sur sa sortie standard (comme `cat`), mais qui n'écrit pas plus vite qu'une certaine vitesse spécifiée en octets par seconde. (À vous de choisir si cette vitesse est codée en dur ou si elle peut être donnée en argument au programme.) Utilisez `alarm()` comme dans l'exercice précédent, et essayez à nouveau de garder le gestionnaire de signal aussi simple que possible.

Pour tester votre programme, vous pouvez, par exemple, exécuter la commande suivante, qui devrait afficher la page man de `strtol()` morceau par morceau (en fonction de la vitesse spécifiée) :

```
$ man strtol | ./slowcat
```

Remarque : Cet exercice est aussi une bonne occasion pour revoir les détails de `read()` et `write()`. En particulier, n'oubliez pas que `write()` peut écrire moins d'octets que ce qui lui a été demandé.