F2B205B - Programmation Système et Réseaux
Partie Unix

C. Lohr

16 Décembre 2009

Modalités

- Document de cours autorisés (polys, sujets de TP, listings de TP)
- Les ordinateurs portables et autres équipements informatiques ne seront pas autorisés dans la salle. Ils devront tout au moins être éteints.
- Réponses à rendre sur copie séparée.
- Durée indicative : 1h

1 Première question

La fonction `system(3)` permet d'exécuter une commande Unix dont le nom et les paramètres sont passés en argument, à la manière de l'exemple suivant : `system("ls -l /usr");`

La fiche (simplifiée) du manuel correspondant à cette fonction vous est fournie en annexe A. C'est votre cahier des charges. Nous vous demandons de proposer votre propre version de cette fonction.

(Ne cherchez pas la perfection de la syntaxe, nous attendons simplement que vous nous montriez que vous avez compris et comment peuvent être utilisés les concepts et les outils vus en cours.)

2 Deuxième question

Dans le cours de programmation socket vous a été présenté l'enchaînement typique d'appels systèmes utilisés pour implémenter un *client/serveur* (ou un *émetteur/récepteur*).

- Pourquoi est-ce que le *serveur* (ou le *récepteur*) utilise l'appel système `bind()` ? Que se passe-t-il s'il ne le fait pas ?
- Pourquoi est-ce que le *client* (ou l'*émetteur*) n'utilise pas l'appel système `bind()` ? Que se passe-t-il s'il le fait ?

3 Troisième question

Le programme fournit en annexe B est syntaxiquement correct (le compilateur ne fait aucune remarque). Il est même fonctionnellement correct (il s'exécute, fait quelque chose). Par contre, d'un point de vu algorithmique il est hautement critiquable. Critiquez !

- Que fait il dans l'état actuel des choses ?
- En quoi est-il mal programmé ?
- Comment arranger les choses ?

Annexe A

SYSTEM(3) Manuel du programmeur

NOM

`system` - Exécuter une commande shell

SYNOPSIS

```
#include <stdlib.h>

int system(const char *command);
```

DESCRIPTION

La fonction **`system()`** exécute la commande indiquée dans *`command`* en appelant `/bin/sh -c` *`command`*, et revient après l'exécution complète de la commande. Durant cette exécution, le signal **`SIGCHLD`** est bloqué, et les signaux **`SIGINT`** et **`SIGQUIT`** sont ignorés.

VALEUR RENVOYÉE

La valeur renvoyée est `-1` en cas d'erreur (par exemple échec de **`fork(2)`**) ou le code de retour de la commande en cas de succès. Ce dernier code est dans le format indiqué dans **`wait(2)`**. Ainsi, le retour de la commande sera `WEXITSTATUS(status)`. Dans le cas où `/bin/sh` ne peut pas être exécuté, le code de retour sera identique à celui d'une commande effectuant un `exit(127)`.

Annexe B

Programme à étudier

```
1  #include <signal.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <string.h>
5
6  void handler(int n) {
7    printf("Signal numero %d\n", n);
8  }
9
10 int main() {
11   for(;;) {
12     signal(SIGQUIT, handler);
13     sleep(60);
14   }
15 }
```