UNIVERSITÉ DE STRASBOURG

Département d'Informatique
LICENCE INFORMATIQUE 2ème année

Programmation Système
TD n° 3

Notions abordées :

- **Exécution de commandes (primitives de la famille `exec*()`)**
- **Duplication de descripteurs de fichier (`dup()` et `dup2()`)**

Exercice 1

Écrire un programme C qui lance la commande `ls` sur un répertoire passé en argument.

Exercice 2

Écrire un programme C qui exécute une commande Unix passée en argument. Exemple :

```
$ ./a.out ls -al
total 320
drwxr-xr-x 2 violard nfs   4096 2011-10-06 15:14 .
drwxr-xr-x 4 violard nfs   4096 2011-10-06 15:04 ..
-rwxr-xr-x 1 violard nfs   7252 2011-10-06 15:14 a.out
...
$
```

Exercice 3

Écrire un programme C qui exécute au plus 3 fois (dans des processus séparés) une commande Unix passée en argument. Si l'une des exécutions provoque une erreur, il ne faut pas réaliser les exécutions suivantes.

Exercice 4

Écrire une fonction (`rediriger_stdout()`) pour rediriger la sortie standard vers un fichier passé comme argument à cette fonction. On considère que le fichier n'existe pas. Écrire une autre fonction pour restaurer la sortie standard (`restaurer_stdout()`). Exemple :

```
int main (int argc, char* argv [])
  {
    printf("avant la redirection\n");
    rediriger_stdout("fichier.out");
    printf("après la redirection\n");
    restaurer_stdout();
    printf("après avoir restauré stdout\n");
    return EXIT_SUCCESS;
  }
```

Dans ce cas, la phrase après la redirection ne sera pas affichée à l'écran mais écrite dans le fichier `fichier.out`.

Exercice 5

Écrire un programme C équivalent à la commande shell suivante :

```
ps aux | grep root | wc -l
```

.