TD Gestion des signaux

Exercice 1

Complétez le code source suivant de manière à ce que le processus ignore tous les signaux.
Rappel : il existe `NSIG` signaux dans le système Unix, avec `NSIG` une constante pré-définie.

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>

int main(void){

  int Nb_Sig;

  /* Code à compléter */

  while(1){
     sleep(5);
  }  /* Le processus attend des signaux   */
  return 0;
}
```

Exercice 2

1. Ecrire un programme qui permet d'ignorer les deux signaux correspondant à la frappe des touches « `CTRL C` » (signal `SIGINT`) et « `CTRL \` » (signal `SIGQUIT`).
2. Ecrire un programme qui réalise un déroutement du signal « `CTRL C` ». Le déroutement consiste à afficher le message « Signal reçu » en indiquant le numéro du signal.
3. Ecrire un programme qui réalise un déroutement du signal « `CTRL C` ». Le déroutement consiste à compter le nombre de signaux reçus et à afficher le message « Signal reçu n fois» en indiquant le numéro du signal. Le $5^{ième}$ « `CTRL C` » termine le processus.

Pour chaque question ci-dessus :

a. utiliser `signal()` pour installer le comportement.
b. utiliser `sigaction()` pour installer le comportement.

Exercice 3

Les signaux `SIGUSR1` et `SIGUSR2` sont deux des signaux standards définis par la norme POSIX. Il s'agit de signaux émis par un processus utilisateur.
Ecrire un programme en C qui permet à un processus :

- de créer un processus fils,
- d'envoyer un signal `SIGUSR1` à son processus fils après avoir testé son existence avec l'appel système approprié,
- d'attendre la terminaison de ce processus fils et de récupérer son « status » à la terminaison.

Exercice 4

Analysez le programme suivant et dire ce qu'il fait.
Remarque : la fonction `execvp()` permet d'exécuter la commande qui lui est transmise en premier argument. Le deuxième argument de `execvp()` indique les arguments disponibles pour la commande à exécuter.

```
1.    #include <stdio.h>
2.    #include <signal.h>
3.    #include <stdlib.h>
4.    #include <unistd.h>

5.    int main(int argc, char **argv)
6.    {
7.      int retour;
8.      struct sigaction mon_action;

9.      if (argc < 2) {
10.         printf("Erreur\n");
11.         exit(1);}

12.     mon_action.sa_handler = SIG_IGN;
13.     sigemptyset(&mon_action.sa_mask);
14.     mon_action.sa_flags = 0;

15.     retour = sigaction(SIGHUP, &mon_action, NULL) ;
16.     if (retour == -1) {
17.         printf("Erreur sigaction\n");
18.         exit(1);}

19.     execvp(argv[1], argv + 1);
20.     printf("Erreur execvp\n");
21.   }
```