

# Info2 #, Programmation système TP5, Signaux première partie

Enseignant : M. Nakechbandi

## Exercice 1

1. Compiler, exécuter et tester le programme `sig3_1.c` (page 11 )
2. Modifier ce programme pour ignorer les cinq premiers CTRL-C (signal `SIGINT`) et réagit au sixième.
3. Idem modifier le programme `sig3_2.c` pour que le programme s'arrête quand on frappe CTRL-\ (signal `SIGQUIT`)

## Exercice 2

1. Compiler, exécuter et tester le programme `sig4_1.c` (page 13 )
2. Modifier ce programme (`ex2.2.c`) pour que le processus exécute 3 handlers sur réception de `SIGINT` de `SIGUSR1` ou de `SIGUSR2`, un message différent selon le signal sera affiché.

## Exercice 3

1. Compiler, exécuter et tester le programme `sig1.c` (page 9 )
2. S'inspirer de ce programme pour écrire un nouveau programme (`ex3_2.c`) prenant un pid en paramètre. Ce programme saisit un caractère au clavier qui peut être '+', '-' ou 'A'. Cette saisie se fait en boucle jusqu'à ce que l'utilisateur tape 'A'. Après chaque saisie et en fonction du caractère tapé par l'utilisateur, le programme envoie les signaux suivants :
  - si le caractère est '+', le programme envoie le signal `SIGUSR1`
  - si le caractère est '-', le programme envoie le signal `SIGUSR2`
  - si le caractère est 'A', le programme envoie le signal `SIGINT`

Ces signaux sont envoyés au processus dont le pid est passé en paramètre.

**Tester ce programme en couple avec le programme de la question 2.2** (`ex3_2.c`)

**Aide :**

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/signal.h>
int main(int argc, char* argv[]) {
    int pid; //pid du pss qui affiche les valeurs
    char ordre;
    pid = atoi(argv[1]);
    do {printf("que voulez-vous faire ? \n");
        ordre = getchar();
        getchar( ); /*pour éliminer le retour-chariot*/
        /*on envoie le bon signal en fonction du caractère saisi*/
```

```

        if (ordre == '+') { kill(... ,...);}
        if (ordre == '-') { kill(... ,...);}
        if (ordre == 'A') { kill(... ,...);}
    }
    while (ordre != 'A');
}

```

3. Tester ce programme en couple avec le programme de la question 2.2 d'un autre utilisateur. Quelle est votre conclusion

#### Exercice 4

1. Ecrire un programme `e(ex4_1.c)` qui
  - affiche par défaut et à l'infini la valeur d'une variable `val` (initialisée à 1)
  - sur réception du signal `CTRL-C`, augmente la valeur de `val` de 1
  - sur réception du signal `CTRL-\`, diminue la valeur de `val` de 1

#### Aide :

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/signal.h>
int val;
void augmenter(int sig) { val++;}
void diminuer(int sig) { val--;}
int main() {
    val = 1;
    signal(..., augmenter);
    signal(..., diminuer);
    while (1)
    {
        printf("val = %d\n", val);
        sleep(1);    } }

```

2. Modifier le programme précédent (`ex4_2.c`) pour qu'il prenne en paramètres les bornes min. et max. Lorsque la borne max. (ou min) est atteinte le programme s'arrête. **Indications :**

- `min` et `max` pouvant être lu comme paramètres. Utiliser :
 

```

if (argc != 3) {printf("Erreur dans le nb d'arguments\n"); exit(1);}
min = atoi(argv[1]);
max = atoi(argv[2]);
...

```
- S'inspirer de l'exemple `sig3_2.c` page 11 pour arrêter le programme avec un traitement standard du signal en utilisant `signal(SIGINT, SIG_DFL)` ...

La primitive `sigaction` est une autre structure décrit le comportement utilisé pour le traitement d'un signal :

```
struct sigaction {
    void (*sa_handler) ();
    sigset_t sa_mask;
    int sa_flags;}
```

- **sa\_handler** : fonction de traitement (ou `SIG_DFL` et `SIG_IGN`) , il est positionné par `sigaction`
- **sa\_mask** : ensemble de signaux supplémentaires à bloquer pendant le traitement
- **sa\_flags** : différentes options

### Exercice 5 :

**5.1/** Analyser puis compiler, exécuter et tester le programme `ex5_1.c` ci-dessous. Tester plusieurs fois en envoyant au processus les signaux `SIGINT` puis `SIGUSR1`. (d'une fenetre `kill -10 pid`)

```
/* Exercice ex5_1.c : */
#include<stdio.h>
#include<sys/types.h>
#include<sys/signal.h>
/* handler (fonction) associé au signal SIGINT */
void traitement (int sig)
{ printf("signal SIGINT reçu !\n"); }
int main()
{
    struct sigaction action;
    /* structure permettant de construire un handler*/
    sigset_t masque;
    /* masque contenant les éventuels signaux masqués*/
    /* pendant l'execution du handler */
    /* on initialise le masque à vide : */
    /* on en souhaite masquer aucun signal */
    sigemptyset(&masque); /* on remplit la structure du handler*/
    action.sa_handler = traitement;
    /* la fonction associée au signal */
    action.sa_mask = masque; /* le masque des signaux bloqués */
    action.sa_flags = 0; /* d'éventuelles options */
    /* on installe le handler */
    /* cela veut dire qu'on l'associe a un signal particulier :ici SIGINT*/
    /* cela veut aussi dire qu'on rend le handler actif */

    sigaction(SIGINT, &action, NULL);
    printf("Mon pid est %d\n", getpid());
    while (1) { printf("un tour! \n"); sleep(1); }
}
```

C'est un programme de test pour l'installation d'un handler de signal en utilisant la primitive `sigaction`. Ce programme boucle à l'infini. Pour l'arrêter, il faut récupérer le pid du processus et le tuer dans une autre fenêtre par un `kill -9 pid`. Pour tester ce programme, il suffit de le lancer et de taper de temps à autres CTRL+C. Au lieu de s'arrêter, le programme affiche le message "signal SIGINT reçu !"

**5.2 /** Idem que l'exercice ex2.2 (de lundi), modifier ce programme (`ex5_2.c`) pour que le processus exécute 3 handlers sur réception de SIGINT de SIGUSR1 ou de SIGUSR2, un message différent selon le signal sera affiché.

**5.3 /** Modifier le programme précédent 5.1 (`ex5_3.c`) afin que le processus ignore le signal SIGUSR1. Faire des tests en envoyant différents signaux à différents moments.

**Indication :** on ajoute :

```
action.sa_handler = SIG_IGN;
sigaction(SIGUSR1, &action, NULL);
```

**Exercice 6 :** Refaire l'exercice 4 de tp de lundi (`ex6.c`) en utilisant la structure `sigaction`. Rappelons que cet exercice :

- affiche par défaut et à l'infini la valeur d'une variable `val` (initialisée à 1)
- sur réception du signal CTRL-C, augmente la valeur de `val` de 1
- sur réception du signal CTRL-\, diminue la valeur de `val` de 1

### Exerce 7 : Utilisation du masque

Modifier le programme de l'exercice 5.1 (`ex7.c`) précédent pour que le processus, au lieu de boucler indéfiniment, attende jusqu'à ce qu'il ait reçu le signal SIGINT (et uniquement celui là). La réception du signal SIGINT doit toujours provoquer l'exécution du handler *traitement*. Tester en envoyant éventuellement une ou plusieurs fois le signal SIGUSR1 avant d'envoyer le signal SIGINT.

**Indication :** On remplace la boucle `while (1)` par

```
sigfillset(&masque);
sigdelset(&masque, SIGINT);
sigsuspend(&masque);
```

**Explication :** On utilise ici la primitive `sigsuspend`, il faut préparer le masque, on met tous les signaux sauf SIGINT, `sigsuspend` bloque le processus en attente de tout signal ne figurant pas dans le masque.