



**Université d'AIN TEMOUCHENT – Algérie**  
Faculté des Sciences et de la Technologie  
Département de Mathématiques et d'Informatique



M1 - CySIA

Matière: **Système d'exploitation Avancée**

**Chapitre : Les signaux**

**TD n° : 02**

**Exercice 01:**

**Partie 1:**

- 1) Écrire un programme qui affiche infiniment et toutes les 2 secondes le message "Bonjour tout le monde!"
- 2) En exécutant le programme précédent, quel sera le comportement par défaut des signaux suivants en vérifiant à chaque fois avec la commande ps :
  - a. SIGINT [Ctrl + C]
  - b. SIGTSTP [Ctrl + Z]
  - c. SIGQUIT [Ctrl + \] (Ctrl + Alt Gr + 8)
- 3) Tuez les processus stoppés avec la commande kill.
- 4) Quelle est la combinaison de touches du signal SIGSTOP, et quelle est la différence entre ce dernier et le signal SIGTSTP ?

**Partie 2:**

Vous avez le programme nommé suivant:

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
int x=0;
void interruption(int signum)
{
    switch (signum)
    {
        case SIGINT:
            printf("\nCTRL-C\n");
            x++;
            break;
        case SIGQUIT:
            printf("\nCTRL-\\ \n");
            x--;
            break;
        default:
            printf("\nAutre signal\n");
            if (!x)
                signal(SIGINT, SIG_DFL);
    }
    printf("x = %d\n",x);
}
int main(void)
{
    signal(SIGINT, interruption); /* Récupération de CTRL-C */
    signal(SIGQUIT, interruption); /* Récupération de CTRL-\\ */
    signal(SIGTSTP, interruption); /* Récupération de CTRL-Z */
    for (;;)
    {
        printf("-"); // On veut vraiment l'imprimer sur l'écran !!!
        fflush(stdout);
        sleep(1);
    }
}
```

```

    }
    return EXIT_SUCCESS;
}

```

- 5) Que se passe-t-il pendant l'exécution de ce programme si on tape :
  - CTRL-C,
  - CTRL-C deux fois,
  - CTRL-\,
  - CTRL-Z
  - CTRL-Z puis CTRL-\ puis CTRL-C ? (Dans une nouvelle exécution)
- 6) Que fait la commande fflush(stdout)?
- 7) Donner plusieurs façons de quitter ce programme.
- 8) Modifier le comportement des signaux suivants :
  - a. SIGINT : Ignorer
  - b. SIGTSTP : Afficher un message et continuer l'exécution du programme.

### Exercice 02:

Considérer le programme suivant:

```

pid_t pid;
void onalarm(); /*handler*/
int main()
{
    pid=fork();
    if (pid==-1) printf("erreur création de processus");
    else
        if (pid==0)
        {
            printf("fils cree %d\n",pid);
            printf("je suis le fils mon pid est %d\n",getpid());
            sleep(2);
            for(;;)
                printf("je boucle !!!!\n");
            exit(0);
        }
    else {
        printf("valeur du fork %d",pid);
        signal(SIGALRM,onalarm);
        alarm(5);
        wait(NULL);
    }
    printf("fin pere\n");
    return 0;
}

void onalarm()
{
    printf("\ntraitement onalarm\n");
    kill(pid,SIGKILL);
}

```

- 1) Que fait ce programme ?
- 2) Vérifier votre réponse sur la machine

### Exercice 03:

- 1) Que fait le programme suivant:

```

int main (int argc, char *argv [])
{
    if (! fork())
    {
        for(int i=0;i<10;i++);        //simule un petit calcul
    }
}

```

```

        exit(1) ;
    }
    while(1);        //Simule un calcul infini
}

```

2) Ajoutez une **fonction Handler** `sigchld()` et le **code nécessaire** afin que le père n'attende jamais son fils de façon bloquante et que le fils ne devienne pas zombie. (*Le signal SIGCHLD est un signal qui est automatiquement envoyé par le fils à son père lorsque le fils se termine (par un exit, un return, ou autre)*).

3) Considérez le programme suivant :

```

void sigintP (int sig)
{
    ...
}

void sigalrm (int sig)
{
    ...
}

void sigintF (int sig)
{
    ...
}

void sigchld (int sig)
{
    ...
}

int main (void)
{
    signal(SIGCHLD, sigchld);
    if (fork()==0)
    {
        signal (SIGINT, sigintF);
        while(1)
        {
            printf ("ici fils \n");
            sleep(1);
        }
    }
    while(1)
    {
        signal (SIGINT, sigintP) ;
        printf ("ici père \n") ;
        sleep(1) ;
    }
    return 0 ;
}

```

Complétez ce code de manière à réaliser les traitements suivants :

- Le père n'attende jamais son fils de façon bloquante et que le fils ne devienne pas zombie
- Si l'utilisateur presse les touches Ctrl+C lorsque le programme s'exécute, les processus père et fils ne se terminent pas immédiatement, mais après un délai de 5 secondes.

**Indication :** Ctrl+C doit mener le fils à effectuer un appel système `alarm(5)` qui envoie automatiquement le signal `SIGALRM` après 5 secondes.

#### Exercice 04:

1) Que fait le programme suivant ?

```

#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>

```

```

#include <signal.h>
#define N 5
/*0*/
int main()
{
    pid_t pid [N] ;
    int i ;
    /*1*/
    for(i=0; i<N; i++)
        if((pid[i]=fork())==0)
        {
            /*2*/
            while(1)
                printf("ici fils %d",i);
        }
    /*3*/
}

```

- 2) Pourquoi il ne s'arrête pas avec Ctrl+c ?
- 3) On veut que le processus père utilise les signaux SIGSTOP et SIGCONT pour suspendre (bloquer) et reprendre (débloquer) l'exécution de ses processus fils:
  - a) Au départ tous les processus fils créés doivent se mettre en pause.
  - b) Après la création de tous les fils, le processus père répète continuellement le traitement suivant en commençant par le premier fils : il envoie le signal SIGCONT à un fils puis s'endort pendant 1 seconde. À son réveil, il envoie SIGSTOP au même fils et SIGCONT au fils suivant (le fils suivant du dernier est le premier).
  - c) Lorsqu'un processus fils reçoit le signal SIGCONT, il affiche le message indiquant qu'il a capturé le signal SIGCONT avant de poursuivre son exécution.

#### **Exercice 05:**

- 1) Que fait le programme suivant:

```

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
    system("stty -echo");
    char input;
    printf("Entrez quelque chose : ");
    scanf("%c", &input);
    system("stty echo");
    printf("\nVous avez entré : %c\n", input);
    return 0;
}

```

- 2) Quel est le rôle de la commande system("stty -echo"); et system("stty echo");?
- 3) Modifier le programme de sorte que si l'utilisateur appuie sur une lettre au clavier, le processus s'envoie lui-même un signal SIGUSR1 qui permet d'afficher le caractère pressé.
- 4) Peut-on utiliser un autre signal ?
- 5) Même question 3°, mais cette fois, le caractère affiché est aléatoire, c.-à-d. si j'appuie sur la lettre 'a' la première fois, il affiche 'h', la deuxième fois, il affiche 's' etc.