Manipulation des ensembles de signaux

- **La nome POSIX fournit une interface standard pour la construction et la manipulation d'ensemble de signaux.**

- **Le type sigset_t correspond à de tels ensembles**

- **<u>Fonctions</u> :**

Fonction	effet
`int sigemptyset(sigset_t *p_ens) ;`	`*p_ens={}`
`int sigfillset(sigset_t *p_ens) ;`	`*p_ens={1,…,NSIG}`
`int sigaddset(sigset_t *p_ens,int sig);`	`*p_ens=*p_ensU{sig}`
`int sigdelset(sigset_t *p_ens,int sig);`	`*p_ens=*p_ens-{sig}`
`int sigismember(sigset_t *p_ens,int sig);`	`sig est dans *p_ens`

- **Chacune de ces fonction renvoie –1 en cas d'erreur, 0 sinon**

- **la fonction** *sigismember* **renvoie 0 ou 1 selon que** *sig* **appartient ou non à** **p_ens*.

Blocage des signaux

- **L'installation d'un masque de blocage des signaux est réalisé par un appel à la fonction :**

```
int sigpromask(  int op,
                 const sigset_t *p_ens,
                 sigset_t *p_ens_ancien)
```

  - **Le nouveau masque est construit à partir de l'ensemble** **p_ens* **et du masque antérieur** **p_ens_ancien*
  - **Le paramètre** *op* **permet de déterminer le nouvel ensemble :**

Valeur de *op*	**Nouveau masque**
SIGSET_TMASK	*p_ens
SIG_BLOCK	*p_ens U *p_ens_ancien
SIG_UNBLOCK	*p_ens_ancien - *p_ens

  - **La valeur de retour de la fonction est 0 en cas de réussite, -1 sinon.**

- **Liste des signaux pendants bloqués**
  Un appel à la fonction :

```
int sigpending(sigset_t *p_ens);
```

  - **Ecrit à l'adresse** *p_ens* **la liste des signaux pendants qui sont bloqués.**

Exemple

```
#include <stdio.h>
#include <signal.h>
sigse_t ens1,ens2 ;
int sig ;
main()
{
/* construction de l'ensemble ens1={ SIGINT, SIGQUIT, SIGUSR1} */
sigemptyset(&ens1) ;
sigaddset(&ens1,SIGINT) ;
sigaddset(&ens1,SIGQUIT) ;
sigaddset(&ens1,SIGUSR1) ;

/*installation du masque */
sigpromask(SIGSET_TMASK,&ens1,NULL) ;

/*mise en sommeil du processus */
sleep(15) ;

/* Extraction des signaux pendants masqués */
sigpending(&ens2) ;
printf('signaux pendants : ') ;
for (sig=1 ;sig<NSIG ;sig++)
     if (sigismember(&ens2,sig))
     printf('%d',sig) ;

/* Déblocage des signaux */

printf('Deblocage des signaux\n') ;
sigemptyset(&ens1) ;
sigpropmask(SIGSET_TMASK,&ens1,NULL) ;
printf('fin du processus') ;
}
```

Manipulation des handlers

❑ La structure sigaction

```
Struct sigaction {
        void (*sa_handler)() ;
        sigset_t sa_mask ; /* signaux à bloquer*/
        int sa_flags ;     /*options*/
        }
```

Le champ *sa_mask* **correspond à une liste de signaux qui doivent être ajoutés, pendant l'exécution du *handler* à ceux déjà bloqués**

Parmi les valeurs de *sa_flags* :

SA_RESTART **: un appel système interrompu par un signal capté est repris au lieu de renvoyer –1**

SA_RESETHAND **: si le signal est capté, il ne sera pas bloqué à sa délivrance et SIG_DFL sera Réinstallé automatiquement**

❑ La primitive sigaction

```
int sigaction( int sig,
        const struct sigaction *p_action,
        struct sigaction *p_action_ancien)
```

Si p_action n'est pas NULL, alors cette primitive permet d'installer la fonction *p_action->sa_handler* **comme *handler* pour le signal sig.**

Les signaux *p_action-> U {sig}* **sont masqués pendant l'exécution du *handler*.**

Exercice : Ecrire une fonction qui lit au clavier une chaîne de caractères. La fonction retourne et renvoie –1 si l'utilisateur n'a pas entré la chaîne de caractères Au bout de 10 secondes.