Signaux

Ensemble de signaux

Cet article présente différents exemples montrant la manipulation des ensembles de signaux, ainsi que le blocage de signaux avec `sigprocmask`.

1 Premier exemple

Le but de ce premier exemple est d'illustrer le fonctionnement de `sigprocmask` qui permet de bloquer des ensembles de signaux. Un premier programme créé un fils qui bloque tous les signaux sauf `SIGINT`. Le père envoie au fils un signal qui est bloqué puis le signal `SIGINT` qui tue le fils.

Le fils commence par bloquer tous les signaux sauf `SIGINT` à l'aide de `sigprocmask`. L'ensemble est rempli à l'aide de `sigfillset` et le signal `SIGINT` est retiré à l'aide de `sigdelset`.

```
sigset_t sigs_new, sigs_old;

sigfillset(&sigs_new);
sigdelset(&sigs_new, SIGINT);
if(sigprocmask(SIG_BLOCK, &sigs_new, &sigs_old) == -1) {
  perror("Erreur lors du blocage des signaux ");
  exit(EXIT_FAILURE);
}
```

Le fils se met ensuite en pause. À noter qu'il ne sortira pas de cette pause car il va recevoir le signal `SIGINT` qui va forcer son arrêt.

```
printf("Fils : je suis insensible à tous les signaux !\n");
pause();
printf("Fils : je suis sorti de ma pause\n");
```

Dans le `main`, le fils est créé à l'aide de `fork`.

```
if((pid = fork()) == -1) {
  perror("Erreur lors de la création du premier fils ");
  exit(EXIT_FAILURE);
}
if(pid == 0)
  fils();
```

Le père envoie dans un premier temps le signal `SIGUSR1` qui sera bloqué.

```
  if(kill(pid, SIGUSR1) == -1) {
    perror("Erreur lors de l'envoi de SIGUSR1 ");
    exit(EXIT_FAILURE);
  }
```

Le père vérifie si le fils s'est arrêté à l'aide de `waitpid`. L'option `WNOHANG` permet de rendre l'appel non bloquant : si aucun fils n'est arrêté, `waitpid` retourne -1. Ce qui est le cas ici.

```
if((tmp = waitpid(pid, NULL, WNOHANG)) == -1) {
    perror("Erreur lors de l'attente de mon fils ");
    exit(EXIT_FAILURE);
  }
  if(tmp != 0)
    printf("Père : bon, je crois que j'ai tué mon fils...\n");
  else
    printf("Père : mon fils est toujours en vie...\n");
```

Le père va ensuite envoyer le signal `SIGINT` pour arrêter le fils et attendre son arrêt avec `waitpid`. Cette fois-ci, le fils s'est bien arrêté.

2 Deuxième exemple

Le but de ce deuxième exemple est d'illustrer le fonctionnement de l'appel `sigpending` qui permet de récupérer l'ensemble des signaux bloqués reçus.

Dans un premier temps, le fils bloque tous les signaux.

```
sigfillset(&sigs_new);
if(sigprocmask(SIG_BLOCK, &sigs_new, &sigs_old) == -1) {
  perror("Erreur lors du blocage des signaux ");
  exit(EXIT_FAILURE);
}
```

Il attend 5 secondes à l'aide de `sleep` puis il récupère les signaux bloqués reçus à l'aide de `sigpending`.

```
if(sigpending(&sigs_bloques) == -1) {
  perror("Erreur lors de la récupération des signaux bloqués ");
  exit(EXIT_FAILURE);
}
```

À l'aide de `sigismember`, il peut vérifier si un signal fait partie de l'ensemble récupéré.

```
if(sigismember(&sigs_bloques, SIGUSR1))
  printf("Fils : j'ai bien reçu SIGUSR1 pendant ma pause.\n");
else
  printf("Fils : je n'ai pas reçu SIGUSR1 pendant ma pause.\n");
```

Le père se contente de créer un fils et de lui envoyer les signaux `SIGUSR1` et `SIGUSR2`. Il attend ensuite sa terminaison à l'aide de `wait`.

3 Troisième exemple

Ce dernier exemple illustre le fonctionnement de `sigsuspend` qui permet d'attendre la réception d'un signal en particulier.

Dans un premier temps, le fils bloque tous les signaux.

```
sigfillset(&sigs_new);
if(sigprocmask(SIG_BLOCK, &sigs_new, &sigs_old) == -1) {
  perror("Erreur␣lors␣du␣blocage␣des␣signaux␣");
  exit(EXIT_FAILURE);
}
```

Il positionne ensuite un gestionnaire sur le signal `SIGUSR2`.

```
sigemptyset(&action.sa_mask);
action.sa_flags = 0;
action.sa_handler = gestionnaire;
if(sigaction(SIGUSR2, &action, NULL) == -1) {
  perror("Erreur␣lors␣du␣positionnement␣du␣gestionnaire␣");
  exit(EXIT_FAILURE);
}
```

Pour attendre des signaux en particulier, il faut créer un ensemble avec tous les autres signaux. Cet ensemble est passé ensuite à `sigsuspend`.

```
sigfillset(&sigs_wait);
sigdelset(&sigs_wait, SIGUSR2);
if(sigsuspend(&sigs_wait) == -1) {
  if(errno != EINTR) {
    perror("Erreur␣lors␣de␣l'attente␣de␣SIGUSR2␣");
    exit(EXIT_FAILURE);
  }
}
```

Le père se contente de créer le fils, de lui envoyer des signaux et d'attendre sa terminaison.

4 Compilation et exécution

Le `makefile` fourni permet de compiler les programmes précédents. Tapez la commande suivante :

```
make
```

Pour tester les programmes, exécutez l'une des commandes suivantes :

```
./exemple1
./exemple2
./exemple3
```