La norme Posix.1

Gestion de signaux

Signaux
Définition, état, action associée *(1)*

- **Notion de signal**
  - Evénement asynchrone
    - interruption terminal (**^C**, **^Z**, **^**)
    - terminaison d'un processus fils...
  - Evénement synchrone (exception)
    - erreur arithmétique (division par 0)
    - violation de protection mémoire...

Signaux
Définition, état, action associée (2)

- **État d'un signal ; action associée**
  - Ignoré (et donc perdu !)
  - Associé à son action par défaut
    - dépend du signal (rien, suspension, reprise, terminaison avec ou sans core...)
  - Associé à une action définie par l'utilisateur (piégé, capturé, « trappé »)
    - « handler » de signal (fonction utilisateur)
- **Signal différé (masqué, bloqué)**
  - Le signal est mémorisé
  - L'action sera effectuée lors du déblocage (démasquage)

Signaux
Liste des signaux de POSIX.1 *(1)*

Signal	Signification	core	Action déf.	Remarque
SIGABRT	abort	✓	fin	
SIGALRM	alarme (time-out)		–	
SIGFPE	exception calcul flottant	✓	fin	exception
SIGHUP	coupure ligne terminal		fin	
SIGILL	instruction illégale	✓	fin	exception
SIGINT	interruption		fin	**^C** au terminal
SIGKILL	terminaison forcée		fin	ni ignorable, ni piégeable, ni blocable
SIGPIPE	écriture sur pipe sans lecteur		fin	exception
SIGQUIT	interruption	✓	fin	**^** au terminal
SIGSEGV	violation mémoire	✓	fin	exception

Signaux
Liste des signaux de POSIX.1 (2)

Signal	Signification	core	Action déf.	Remarque
SIGTERM	terminaison normale		fin	
SIGUSR1	signal utilisateur		—	
SIGUSR2	signal utilisateur		—	
SIGCHLD	changement état d'un fils		—	
SIGSTOP	suspension forcée		**suspension**	ni ignorable, ni piégeable, ni blocable
SIGCONT	reprise		**reprise**	(**fg** et **bg**)
SIGTSTP	suspension douce		**suspension**	**^Z** au terminal
SIGTTIN	lecture terminal depuis un processus background		**suspension**	exception
SIGTTOUT	écriture terminal depuis un processus background		**suspension**	exception

Signaux
Liste des signaux de POSIX.1 (3)

- Aucune priorité entre les différents signaux
- L'ordre de délivrance de plusieurs signaux « simultanés » n'est pas garantie

Délivrance d'un signal à un processus

- Caractères spéciaux au terminal
  - ^C, ^Z, ^\, ...
- Fonctions spéciales du shell
  - fg, bg, kill...
- Primitive Posix : fonction `kill()`

```
#include <sys/types.h>
#include <signal.h>
int kill(pid_t pid, int sig)
```

Etat d'un signal en ANSI C
Fonction `signal()` *(1)*

```
#include <signal.h>
void (*signal(int sig, void (*ph)(int)))(int);
```

ou, si l'on préfère

```
typedef void (*Ptr2Handler)(int);
Ptr2Handler signal(int sig, Ptr2Handler ph);
```

- Positionne l'action associée à la réception du signal **`sig`**
- L'action associée est **`ph`** (« pointer to handler »)
  - **`SIG_IGN`** : signal ignoré
  - **`SIG_DFL`** : action par défaut
  - une fonction utilisateur (paramètre **`int`**, retour **`void`**) : piégé
- Retourne l'ancienne action associée

Etat d'un signal en ANSI C
Fonction `signal()` (2)

```
#include <signal.h>

void on_signal(int sig) {
 printf("*** signal %d\n", sig);
}

main() {
 void (*ph)(int);

 signal(SIGQUIT, SIG_IGN);
 ph = signal(SIGINT, SIG_IGN);
 printf("INT et QUIT ignorés\n");
 sleep(5);
 signal(SIGQUIT, on_signal);
 signal(SIGINT, on_signal);
 printf("INT et QUIT trappés\n");
 sleep(5);

 signal(SIGQUIT, SIG_DFL);
 signal(SIGINT, ph);
 printf("INT restauré "
        "QUIT défaut\n");
 sleep(5);
}
```

Etat d'un signal en ANSI C
Fonction `signal()` *(3)*

```
% test-signal
INT et QUIT ignorés
^\^CINT et QUIT trappés
^\*** signal 3
^\*** signal 3
^C*** signal 2
INT restauré QUIT défaut
^C
%
```

Etat d'un signal en Posix

- **Inconvénients des signaux d'Ansi C**
  - Impossibilité de consulter l'action/état courant(e)
  - Impossibilité de bloquer (masquer) d'autres signaux pendant l'exécution du handler
  - Pas de possibilité d'extension
- **Posix introduit de nouveaux mécanismes**
  - Blocage (masquage) de signaux (emprunté à BSD)
  - Fonction **`sigaction()`** comme remplacement de **`signal()`**

Etat d'un signal en Posix
Action associée

	Bloqué	**Débloqué**
Ignoré	rien	rien
Associé à l'action par défaut	action différée au déblocage	action immédiate
Piégé	exécution du *handler* différée au déblocage	exécution du *handler* immédiate

Etat d'un signal en Posix
Masque des signaux *(1)*

```
#include <signal.h>
int sigprocmask(int how,
                const sigset_t *set,
                sigset_t *old_set);
```

- **set** contient l'ensemble des signaux à masquer ou démasquer
- how détermine la fonction à effectuer
  - **SIG_BLOCK** : bloque les signaux de **set**
  - **SIG_UNBLOCK** : débloque les signaux de **set**
  - **SIG_SETMASK** : positionne le masque du processus à **set**
- **old_set** contient l'ancien masque

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

- **sigpending** retourne les signaux bloqués en attente

Etat d'un signal en Posix
Masque des signaux (2)

- **Ensemble de signaux**
  - Ensemble de bits, 1 bit par signal
- **Fonctions de manipulation**

```
#include <signal.h>
int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);
int sigaddset(sigset_t *set, int sig);
int sigdelset(sigset_t *set, int sig);
int sigismember(const sigset_t *set,
                int sig);
```

État d'un signal en Posix
Fonction `sigaction()` *(1)*

```
#include <signal.h>
int sigaction(int sig,
              const struct sigaction
*actp,
              struct sigaction *old_actp);
```

- Champs de `struct sigaction`
  - **`void (*sa_handler)(int)`** fonction de capture (identique à **`signal()`**)
  - **`sigset_t sa_mask`** masque des signaux à bloquer lors de l'exécution du handler
  - **`int sa_flags`** utile seulement pour **`SIGCHLD`**

État d'un signal en Posix
Fonction `sigaction()` (2)

```
#include <signal.h>

void on_signal(int sig) {
 printf("*** signal %d\n", sig);
 sleep(5);
 printf("*** fin handler\n");
}

main() {
 struct sigaction sigact;
 sigset_t msk_int, msk_quit;

 sigemptyset(&msk_int);
 sigaddset(&msk_int, SIGINT);
 sigemptyset(&msk_quit);
 sigaddset(&msk_quit, SIGQUIT);
 sigact.sa_handler = on_signal;
 sigact.sa_mask = msk_quit;
 sigaction(SIGINT, &sigact,
           NULL);
 sigact.sa_mask = msk_int;
 sigaction(SIGQUIT, &sigact,
           NULL);

 printf("INT et QUIT trappés\n");
 sleep(10);
}
```

État d'un signal en Posix
Fonction `sigaction()` *(3)*

```
% test-sigaction
  INT et QUIT trappés
  ^C*** signal 2
  ^\^\*** fin handler
  *** signal 3
  *** fin handler
%
```

État d'un signal en POSIX
Durée de vie du handler

- **Lorsque la fonction `sigaction()` est utilisée pour trapper un signal, le handler est valide jusqu'à ce qu'un prochain `sigaction()` l'invalide**
- **En revanche la durée de vie du handler établi par `signal()` est dépendante de l'implémentation**
  - après réception du signal, l'action par défaut est rétablie
  - on est donc souvent conduit à réarmer le *handler* dans le *handler* lui-même (cas d'UNIX SVR4, de SOLARIS...)