NOM

sigaction, sigprocmask, sigpending, sigsuspend - Fonctions POSIX de manipulations de signaux.

SYNOPSIS

```
#include <signal.h>

int sigaction (int signum, const struct sigaction * act, struct sigaction *oldact);

int sigprocmask (int how, const sigset_t * set, sigset_t * oldset);

int sigpending (sigset_t * set);

int sigsuspend (const sigset_t * mask);
```

DESCRIPTION

L'appel système sigaction sert à modifier l'action effectuée par un processus à la réception d'un signal spécifique.

signum indique le signal concerné, à l'exception de SIGKILL et SIGSTOP.

Si act est non nul, la nouvelle action pour le signal signum est définie par act. Si oldact est non nul, l'ancienne action est sauvegardée dans oldact.

La structure sigaction est définie par quelque chose comme :

```
struct sigaction {
    void     (* sa_handler)   (int);
    void     (* sa_sigaction) (int, siginfo_t *, void *);
    sigset_t    sa_mask;
    int         sa_flags;
    void     (* sa_restorer)  (void);
}
```

Sur certaines architectures on emploie une union, il ne faut donc pas utiliser ou remplir simultanément sa_handler et sa_sigaction.

L'élément sa_restorer est obsolète et ne doit pas être utilisé, POSIX ne mentionne pas de membre sa_restorer.

sa_handler indique l'action affectée au signal signum, et peut être SIG_DFL pour l'action par défaut, SIG_IGN pour ignorer le signal, ou un pointeur sur une fonction de gestion de signaux.

sa_mask fournit un masque de signaux à bloquer pendant l'exécution du gestionnaire. De plus le signal ayant appelé le gestionnaire est bloqué à moins que les attributs SA_NODEFER ou SA_NOMASK soient précisés.

sa_flags spécifie un ensemble d'attributs qui modifient le comportement du gestionnaire de signaux. Il est formé par un OU binaire (|) entre les options suivantes :

SA_NOCLDSTOP
: Si signum vaut SIGCHLD, ne pas recevoir les signaux de notification d'arrêt d'un processus fils (quand le fils reçoit un signal SIGSTOP, SIGTSTP, SIGTTIN ou SIGTTOU).

SA_ONESHOT ou SA_RESETHAND
: Rétablir l'action à son comportement par défaut une fois que le gestionnaire a été appelé (C'est le comportement par défaut avec la fonction signal(2))

SA_RESTART
Fournir un comportement compatible avec la sémantique BSD en redémarrant automatiquement les appels systèmes lents interrompus par l'arrivée du signal.

SA_NOMASK ou SA_NODEFER
Ne pas empêcher un signal d'être reçu depuis l'intérieur de son propre gestionnaire.

SA_SIGINFO
Le gestionnaire de signal recevra trois arguments, et non plus un seul. Dans ce cas, il faut utiliser le membre sa_sigaction et non pas sa_handler. (Le champ sa_sigaction est apparu dans Linux 2.1.86.)

Le paramètre siginfo_t de la routine sa_sigaction est une structure contenant les éléments suivants :

```
siginfo_t {
        int     si_signo;       /* Numéro de signal         */
        int     si_errno;       /* Numéro d'erreur          */
        int     si_code;        /* Code du signal           */
        pid_t   si_pid;         /* PID de l'émetteur        */
        uid_t   si_uid;         /* UID réel de l'émetteur   */
        int     si_status;      /* Valeur de sortie         */
        clock_t si_utime;       /* Temps utilisateur écoulé */
        clock_t si_stime;       /* Temps système écoulé     */
        sigval_t si_value;      /* Valeur de signal         */
        int     si_int;         /* Signal Posix.1b          */
        void *  si_ptr;         /* Signal Posix.1b          */
        void *  si_addr;        /* Emplacement d'erreur     */
        int     si_band;        /* Band event               */
        int     si_fd;          /* Descripteur de fichier   */
}
```

Les champs si_signo, si_errno and si_code sont définis pour tous les signaux. Le reste de la structure peut être une union, et il ne faut donc tenir compte que des champs qui sont significatifs pour le signal reçu. L'appel-système kill(2), les signaux Posix.1b et SIGCHLD remplissent les champs si_pid et si_uid. SIGCHLD remplit aussi si_status, si_utime et si_stime. si_int et si_ptr sont fournis par l'émetteur d'un signal Posix.1b. SIGILL, SIGFPE, SIGSEGV et SIGBUS remplissent si_addr avec l'adresse de l'erreur. SIGPOLL remplit si_band et si_fd.

si_code indique la raison pour laquelle le signal a été émis. Il s'agit d'une valeur, pas d'un masque de bits. Les valeurs possibles pour tous les signaux sont les suivantes :

si_code	
Valeur	Origine du signal
SI_USER	kill, sigsend ou raise
SI_KERNEL	Noyau
SI_QUEUE	sigqueue
SI_TIMER	Fin d'un time
SI_MESGQ	Changement d'état mesq
SI_ASYNCIO	Fin d'une AIO
SI_SIGIO	SIGIO empilé

SIGILL	
ILL_ILLOPC	opcode illégal
ILL_ILLOPN	opérande illégale
ILL_ILLADR	mode ’dadressage illégal
ILL_ILLTRP	trappe illégale
ILL_PRVOPC	opcode priviliégié
ILL_PRVREG	registre privilégié
ILL_COPROC	erreur de coprocesseur
ILL_BADSTK	erreur interne de pile

SIGFPE	
FPE_INTDIV	division entière par zéro
FPE_INTOVF	débordement entier
FPE_FLTDIV	division réelle par zéro
FPE_FLTOVF	débordement réel
FPE_FLTUND	débordement inférieur réel
FPE_FLTRES	résultat réel inexact
FPE_FLTINV	opération réelle invalide
FPE_FLTSUB	indice hors intervalle

SIGSEGV	
SEGV_MAPERR	adresse sans objet
SEGV_ACCERR	permissions invalides

SIGBUS	
BUS_ADRALN	alignement d’adresse invalide
BUS_ADRERR	adresse physique inexistante
BUS_OBJERR	erreur matérielle spécifique

SIGTRAP	
TRAP_BRKPT	point d’arrêt du processus
TRAP_TRACE	suivi d’exécution du processus

SIGCHLD	
CLD_EXITED	fils terminé normalement
CLD_KILLED	fils tué par un signal
CLD_DUMPED	fils terminé anormalement
CLD_TRAPPED	fils en cours de suivi
CLD_STOPPED	fils arrêté
CLD_CONTINUED	fils arrêté a redémarré

SIGPOLL	
POLL_IN	données disponibles en entrée
POLL_OUT	buffers de sortie libres
POLL_MSG	message disponible en entrée
POLL_ERR	erreur d'entrée/sortie
POLL_PRI	entrée haute priorité disponible
POLL_HUP	périphérique débranché

L'appel sigprocmask est utilisé pour changer la liste des signaux actuellement bloqués. Son comportement est dépendant de la valeur de how, avec les conventions suivantes :

SIG_BLOCK
: L'ensemble des signaux bloqués est l'union de l'ensemble actuel et de l'argument set.

SIG_UNBLOCK
: Les signaux dans l'ensemble set sont supprimés de la liste des signaux bloqués. Il est possible de débloquer un signal non bloqué.

SIG_SETMASK
: L'ensemble des signaux bloqués est égal à l'argument set.

Si oldset est non nul, la valeur précédente du masque de signaux est stockée dans oldset.

L'appel sigpending permet l'examen des signaux en attente (qui se sont déclenchés en étant bloqués). Le masque de signaux en attente est stocké dans set.

L'appel sigsuspend remplace temporairement le masque de signaux bloqués par celui fourni dans mask puis endort le processus jusqu'à arrivée d'un signal.

VALEUR RENVOYÉE

sigaction, sigprocmask, et sigpending renvoient 0 s'ils réussissent, ou -1 s'ils échouent, auquel cas errno contient le code d'erreur. La fonction sigsuspend renvoit toujours -1, avec en principe l'erreur EINTR.

ERREURS

EINVAL Un signal invalide est indique. Ceci se produit également si l'on tente de modifier l'action associée à SIGKILL ou SIGSTOP.

EFAULT act, oldact, set, oldset ou mask pointent en-dehors de l'espace d'adressage accessible.

EINTR L'appel système a été interrompu.

NOTES

Il est impossible de bloquer SIGKILL or SIGSTOP avec l'appel sigprocmask. Les tentatives seront ignorées silencieusement.

Suivant POSIX, le comportement d'un processus est indéfini après qu'il ait ignoré un signal SIGFPE, SIGILL, ou SIGSEGV qui n'avait pas été engendré par une fonction kill() ou raise(). La division entière par zéro a un résultat indéfini. Sur certaines architectures, cela déclenchera un signal SIGFPE. (De même diviser l'entier le plus négatif par -1 peut déclencher SIGFPE). Ignorer ce signal peut mener à des boucles sans fin.

POSIX (B.3.3.1.3) désapprouve le positionnement de SIGCHLD à SIG_IGN. Les comportements BSD et SYSV diffèrent, faisant échouer sous Linux les logiciels BSD qui positionne l'action de SIGCHK à SIG_IGN.

Les spécifications POSIX définissent seulement SA_NOCLDSTOP. L'utilisation des autres options de sa_flags n'est pas portable.

L'option SA_RESETHAND est compatible avec l'option SVr4 du même nom.

L'option SA_NODEFER est compatible avec l'option SVr4 du même nom pour les noyaux 1.3.9 et ultérieurs. Pour les noyaux plus anciens, Linux autorisera la réception de tous les signaux et pas seulement celui qui vient de se déclencher (écrasant effectivement sa_mask).

Les noms SA_RESETHAND et SA_NODEFER pour compatibilité avec SVr4 ne sont présents que dans les bibliothèques 3.0.9 et suivantes.

L'attribut SA_SIGINFO est précisé par POSIX.1b. Son support fut ajouté dans Linux 2.2.

sigaction peut être appelé avec un second argument null pour obtenir le gestionnaire de signaux actuel. On peut aussi vérifier si un signal est valide sur la machine actuelle en l'appellant avec les deuxième et troisième arguments nuls.

Voir sigsetops(3) pour les détails concernant les ensembles de signaux.

CONFORMITÉ

POSIX, SVr4. SVr4 ne documente pas la condition d'erreur EINTR.

NON DOCUMENTÉ

Avant l'introduction de l'attribut SA_SIGINFO il était déjà possible d'obtenir des informations supplémentaires dans le gestionnaire de signal, en lui ajoutant un argument de type struct sigcontext. On peut retrouver ceci dans les sources du noyau. Ce mécanisme est désormais obsolète.

VOIR AUSSI

kill(1), kill(2), killpg(2), pause(2), raise(3), siginterrupt(3), signal(2), signal(7), sigsetops(3), sigvec(2)

TRADUCTION

Christophe Blaess, 1997.