Fiche d’utilisation détaillée de l’appel système SIGACTION

SYNOPSIS

```
#include <signal.h>

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

DESCRIPTION

L'appel système **sigaction**() sert à modifier l'action effectuée par un processus lors de 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 :

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

Il ne faut **surtout pas utiliser ou remplir simultanément** *sa_handler* et *sa_sigaction* car c’est peut-être une union qui les définit.

Le champ *sa_restorer* est obsolète et ne doit pas être utilisé.

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 vers une fonction de gestion de signaux. Cette fonction reçoit le numéro de signal comme seul argument.

Si **SA_SIGINFO** est indiqué dans *sa_flags*, *sa_sigaction* (plutôt que *sa_handler*) doit indiquer la fonction gestionnaire de signaux pour *signum*. Cette fonction prend le numéro du signal comme premier argument, un pointeur sur un *siginfo_t* comme second argument et un pointeur sur un *ucontext_t* (transtypé en void *) comme troisième argument (On n’aura pas l’utilité de ce troisième argument par la suite).

sa_mask fournit un masque des signaux à bloquer durant l'exécution du gestionnaire. Le signal ayant appelé le gestionnaire **est bloqué systématiquement** à moins que l'attribut **SA_NODEFER** ne soit précisé.

sa_flags spécifie un ensemble d'attributs qui modifient le comportement du signal. Il est formé par un **OU binaire « | »**) entre les options suivantes :

SA_NOCLDSTOP
: Cet attribut n'a de sens que lors de la mise en place d'un gestionnaire pour **SIGCHLD**. Indique de ne pas recevoir les signaux de notification lors de l'arrêt d'un processus fils lorsque celui-ci reçoit un signal **SIGSTOP**, **SIGTSTP**, **SIGTTIN** ou **SIGTTOU** ou de reprise lorsque il reçoit **SIGCONT** (voir **wait**(2)).

SA_NOCLDWAIT (Depuis Linux 2.6)
: Cet attribut n'a de sens que lors de la mise en *place* d'un gestionnaire pour **SIGCHLD**, ou lors de la configuration de la disposition de signal de **SIG_DFL**. Indique de ne pas transformer les fils en zombies lorsqu'ils se terminent. Voir aussi **waitpid**(2). Si l'attribut **SA_NOCLDWAIT** a été défini lors de la mise en place d'un gestionnaire pour **SIGCHLD**, la norme POSIX.1 ne spécifie pas si le signal **SIGCHLD** doit être généré lorsqu'un processus fils se termine ; mais sous Linux, le signal **SIGCHLD** est généré dans ce cas.

SA_RESETHAND

Rétablir l'action à son comportement par défaut une fois que le gestionnaire a été invoqué. Cet attribut n'a de sens que lors de la mise en place d'un gestionnaire de signal.
SA_ONESHOT est un synonyme non standard et obsolète pour cet attribut.

SA_ONSTACK

Appeler le gestionnaire avec une pile différente fournie par **sigaltstack**(2). Si cette pile n'est pas disponible, on utilisera la pile par défaut. Cet attribut n'a de sens que lors de la mise en place d'un gestionnaire de signal.

SA_RESTART

Fournir un comportement compatible avec la sémantique BSD en redémarrant automatiquement les appels système lents interrompus par l'arrivée du signal. Cet attribut n'a de sens que lors de la mise en place d'un gestionnaire de signal. Voir **signal**(7) pour une discussion sur le redémarrage d'un appel système.

SA_NODEFER

Ne pas empêcher la réception depuis l'intérieur de son propre gestionnaire du signal associé à ce gestionnaire. Cet attribut n'a de sens que lors de la mise en place d'un gestionnaire de signal.
Attention cet attribut est nécessaire si on sort du gestionnaire d'un signal via **longjmp**(3) et qu'on souhaite pouvoir prendre en compte de nouvelles occurrences de ce signal.
SA_NOMASK est un synonyme non standard et obsolète pour cet attribut.

SA_SIGINFO (Depuis Linux 2.2)

Le gestionnaire de signal recevra trois arguments, et non plus un seul. Dans ce cas, il faudra utiliser le membre *sa_sigaction* et non pas *sa_handler*. Cet attribut n'a de sens que lors de la mise en place d'un gestionnaire de signal.

Le paramètre *siginfo_t* du sous-programme *sa_sigaction* est une structure qui contient 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* et *si_code* sont définis pour l'ensemble des signaux (mais *si_errno* n'est pas utilisé sous Linux).
Le reste de la structure **peut être une union.** En conséquence, il ne faut tenir compte que des champs qui sont significatifs pour le signal reçu :

- Les signaux POSIX.1b et **SIGCHLD** remplissent les champs *si_pid* et *si_uid*.
- **SIGCHLD** remplit également *si_status*, *si_utime* et *si_stime*.
- *si_int* et *si_ptr* sont fournis par l'émetteur d'un signal POSIX.1b. Voir **sigqueue**(2) pour plus de détails.
- **SIGILL**, **SIGFPE**, **SIGSEGV** et **SIGBUS** remplissent *si_addr* avec l'adresse de l'erreur.
- **SIGPOLL** remplit *si_band* et *si_fd*.

Le champ *si_code* est une valeur (pas un masque de bits) indiquant la raison pour laquelle le signal a été émis. La liste suivante indique les valeurs que peut prendre *si_code* pour n'importe quel signal, avec la raison pour laquelle le signal a été généré.

SI_USER
: **kill**(2) ou **raise**(3)

SI_KERNEL
: Envoyé par le noyau.

SI_QUEUE
: **sigqueue**(2)

SI_TIMER
: Fin d'une temporisation POSIX

SI_MESGQ
: Changement d'état d'une file de message POSIX (depuis Linux 2.6.6) ; voir **mq_notify**(3)

SI_ASYNCIO
: Fin d'une AIO

SI_SIGIO
: SIGIO empilé

SI_TKILL
: **tkill**(2) ou **tgkill**(2) (depuis Linux 2.4.19)

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **SIGILL** :

ILL_ILLOPC
: opcode illégal

ILL_ILLOPN
: opérande illégal

ILL_ILLADR
: mode d'adressage illégal

ILL_ILLTRP
: trappe illégale

ILL_PRVOPC
: opcode privilégié

ILL_PRVREG
: registre privilégié

ILL_COPROC
: erreur de coprocesseur

ILL_BADSTK
: erreur interne de pile

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **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

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **SIGSEGV** :

SEGV_MAPERR
: adresse sans objet

SEGV_ACCERR
: permissions invalides pour l'objet

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **SIGBUS** :

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

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **SIGTRAP** :

TRAP_BRKPT
point d'arrêt du processus
TRAP_TRACE
suivi d'exécution du processus

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **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 (depuis Linux 2.6.9)

Les valeurs suivantes peuvent être placées dans *si_code* par un signal **SIGPOLL** :

POLL_IN
données disponibles en entrée
POLL_OUT
tampons de sortie libres
POLL_MSG
message disponible en entrée
POLL_ERR
erreur d'entrées-sorties
POLL_PRI
entrée haute priorité disponible
POLL_HUP
périphérique débranché

VALEUR RENVOYÉE

sigaction() renvoie 0 s'il réussit et -1 s'il échoue.

ERREURS (quand valeur renvoyée = -1)

EFAULT
act ou *oldact* pointent en-dehors de l'espace d'adressage accessible.
EINVAL
Un signal invalide est indiqué. Se produit également si l'on tente de modifier l'action associée aux signaux **SIGKILL** et **SIGSTOP**, qui ne peuvent pas être interceptés ou ignorés.