**Signaux**

* un signal est une interruption logicielle asynchrone d’un processus
* peut être aussi vu comme une forme primitive de communication inter-processus
* le signal peut être envoyé par un processus ou par le noyau

**Opérations :**

1. envoi d'un signal
2. traitement par défaut lors de la réception d'un signal
3. association d’un traitement à la réception d'un signal
4. masquage d'un signal

**Signaux envoyés par le noyau:**

**En cas d'erreur:**

* SIGSEGV: segmentation fault; envoyé par le noyau en cas d'adresse illégale
* SIGBUS : bus error
* SIGFPE : floating-point arithmetic error

**Pour des traitements spéciaux:**

* SIGIO : entrée-sortie asynchrone
* SIGALARM : expiration de timer

**Pour avertir de l'état d'autres processus:**

* SIGCHLD : envoyé par le noyau lorsque l'état d'un processus-fils change

**En cas de dépassement de quotas:**

* SIGXCPU : dépassement de temps CPU
* SIGXFSZ : dépassement de la taille max d'un fichier

**Signaux envoyés depuis le terminal:**

* SIGINT: envoyé si le caractère Control-C est entré au terminal. Provoque la terminaison du processus Signal envoyé par un autre processus:
* SIGKILL: provoque la terminaison du processus. Attention: il faut utiliser les noms symboliques des signaux, les valeurs entières associées sont dépendantes de l’implémentation (ex: SIGCHLD = 17 sous LINUX, 18 sous Solaris et 20 sous FreeBSD)

**Envoi d'un signal par un processus**

int kill (pid\_t pid, int sig);

Envoie le signal sig à un processus ou à un groupe de processus:

* au processus pid si pid > 0
* au groupe du processus appelant si pid = 0
* à tous les processus si pid = -1 (non-conforme POSIX)
* au groupe du processus | pid | si pid < -1

Les mécanismes de protection interviennent pour contrôler la transmission des signaux. La transmission à un processus q d’un signal envoyé par p ne se fera que si:

* UIDeff(p) = UIDeff (q) ou
* UIDreel(p) = UIDreel(q) ou
* UIDeff(p) = root

**Traitements par défaut d'un signal**

Cinq traitements par défaut possibles:

* exit : provoque la terminaison du processus

exemples : SIGINT, SIGKILL, SIGALRM

* core : sauvegarde l'état de la mémoire et termine le processus

exemples : SIGFPE, SIGBUS, SIGSEGV

* stop : suspend l'exécution du processus

exemple : SIGSTOP

* ignore : le signal est ignoré

exemple : SIGCHLD

* continue : le processus suspendu reprend son exécution ou le signal est ignoré

exemple: SIGCONT

**Associer un traitement explicite à un signal**

Se fait par l'association d'un handler (qui détermine la fonction à appeler lors de la réception d'un signal) signal (int sig, [fct]);

* définit la fonction fct comme étant à appeler lors de la réception du signal sig. A son retour, le contrôle retourne à l'endroit où le processus a été interrompu
* le traitement par défault de SIGKILL et SIGSTOP ne peut être modifié signal(sig, SIG\_IGN)
* permet d'ignorer le signal sig signal(sig, SIG\_DFL)
* rétablit le traitement par défaut du signal sig

note : OSIX.4 remplace signal( ) par l'appel sigaction( )

**Ensembles des 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

|  |  |
| --- | --- |
| 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 fonctions renvoie –1 en cas d’erreur, 0 sinon la fonction sigismember renvoie 0 ou 1 selon que sig appartient ou non à \*p\_ens.

**Masquage des signaux**

sigblock( ) -> permet de masquer un signal. Un signal masqué (ou bloqué) est mis en attente. Le signal sera reçu après avoir été débloqué

sigprocmask( ) -> permet de débloquer un signal. Les signaux SIGKILL et SIGSTOP ne peuvent être masqués

Le traitement d'un signal sig se déroule donc de la manière suivante:

* si sig est ignoré, alors le signal est délivré, mais aucune exécution ne s’ensuit
* si sig est masqué, il est ajouté à l'ensemble des signaux en attente d'être traités
* sinon le signal est reçu, et le traitement (défaut ou explicite) est exécuté

Remarque: les fonctions de traitement des signaux ne sont pas standard (entre SVR4 et BSD)