

Les signaux

Définition :

un signal est une interruption logicielle asynchrone

Génération d'un signal.

- CTRL ^ D
- Exception matérielle division par 0
- fonction kill()
- commande kill()
- logiciel

Quelques signaux :

- | | | |
|--|--|-----------------------|
| - SIGABRT ✗ | - SIGALRM | |
| - SIGCHLD | - SIGCONT |] reprendre processus |
| - SIGINT ✗ | - SIGKILL ✗ | |
| - SIGTERM ✗ | - SIGSEGV ✗ | |
| - SIGUSR1 | - SIGSTOP |] suspendre processus |
| - SIGUSR2 | - SIGHUP ✗ | |
| - SIGBUS ✗ | - SIGPIPE ✗ | |

✗ = provoque terminaison du processus

31

Mécanisme :

le traitement de signal s'affectue de la manière suivante :

```
void (* signal (int signal, void (* func) (int))) (int)
```

↳ signal modifie le traitement du signal signal par la fonction func

↳ func peut prendre les valeurs :

- SIG_DEF : traitement par défaut
- SIG_IGN : ignore les signaux
- f : Appel de f à la réception du signal

pt



signal n'appelle pas la fonction
la fonction est appelée quand le signal est reçu



le déroulement vers une fonction à l'aide de
la fonction signal n'est effectué qu'une fois

Exemple : Ecrire un programme qui compte le
nombre de signaux SIGUSR1 reçus

```
int main (int argc, char * argv[]) {  
static int cpt = 0;  
void handler (int s) { (cpt) { cpt++ };  
while (signal (s, handler);  
exit (cpt++);  
printf ("cpteurs %i\n", cpt);  
return;  
}  
int main (int argc, char * argv[]) {  
signal (SIGUSR1, handler);  
while (1) {  
pause();  
}  
exit(0);  
}
```


Envoyer un signal :

- `int kill (pid_t pid, int sig)`
↳ envoie le signal sig au processus pid
- `int raise (int sig)`
↳ \Leftrightarrow `kill (getpid(), sig)`
- `int alarm (int nsec)`
↳ envoie un signal SIGALRM après nsec secondes
- `void pause (void)`
↳ suspend l'exécution du processus jusqu'à la prochaine réception du signal

Exemple : Ecrire un programme qui affiche * par seconde écoulée :

```
void handler (int s) {  
    signal (s, handler);  
    printf ("* ");  
    return;  
}  
  
int main ( ... ) {  
    signal (SIGALRM, handler);  
    while (1) {  
        alarm (1);  
        pause ();  
    }  
    exit(0);  
}
```

✓
mais on
risque de
ne pas être
synchro


```

void handler (int s) {
    signal(s, handler);
    alarm(1);
    printf("x\n");
    return;
}

int main (...) {
    signal(SIGALRM, handler);
    alarm(1);
    while (1) {
        pause();
    }
    exit(0);
}

```

Ensembles de signaux:

Unix permet la gestion d'ensembles de signaux

- sigset_t mysig;
- int sigemptyset (sigset_t * set);
 - ↳ efface l'ensemble des signaux set
- int sigfillset (sigset_t * set);
 - ↳ ajoute tout les signaux à l'ensemble set
- int sigaddset (sigset_t * s, int sig);
 - ↳ ajoute le signal sig à l'ensemble s
- int sigdelset (sigset_t * s, int sig);
 - ↳ supprime le signal sig de l'ensemble s
- int sigismember (const sigset_t * s, int sig)
 - ↳ retourne 1 si sig appartient à s, 0 sinon