I2180 PROGRAMMATION SYSTÈME:

LES SIGNAUX (SUITE)

alarm

```
#include <unistd.h>
unsigned int alarm (unsigned int seconds);
```

- Où: seconds: durée de temporisation avant émission d'un signal SIGALRM au processus appelant
- Il n'y a qu'une temporisation alarm par processus
- Si seconds > 0 : définit une nouvelle temporisation (en annulant éventuellement la précédente)
- Si seconds = 0: annule la temporisation en cours
- Renvoie le nombre de secondes restantes de l'alarme précédemment programmée ; 0 si aucune alarme n'était en cours

Table pour chaque processus:

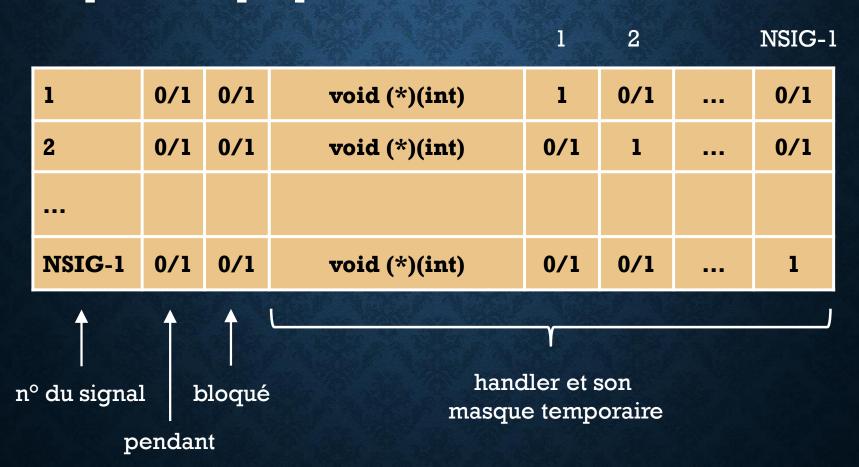


Table pour chaque processus:

				1	2	NSIG-1
1	0/1	0/1	void (*)(int)	1	0/1	 0/1
2	0/1	0/1	void (*)(int)	0/1	1	 0/1
NSIG-1	0/1	0/1	void (*)(int)	0/1	0/1	 1

ensemble des signaux définis (correspondant à sigset_t)

Table pour chaque processus:

				1	2		NSIG-1
1	0/1	0/1	void (*)(int)	1	0/1		0/1
2	0/1	0/1	void (*)(int)	0/1	1		0/1
NSIG-1	0/1	0/1	void (*)(int)	0/1	0/1	•••	1

masque de signaux du processus (défini par *sigprocmask*)

Table pour chaque processus:

	N. A.			1	2		NSIG-1
1	0/1	0/1	void (*)(int)	1	0/1	•••	0/1
2	0/1	0/1	void (*)(int)	0/1	1		0/1
NSIG-1	0/1	0/1	void (*)(int)	0/1	0/1	•••	1

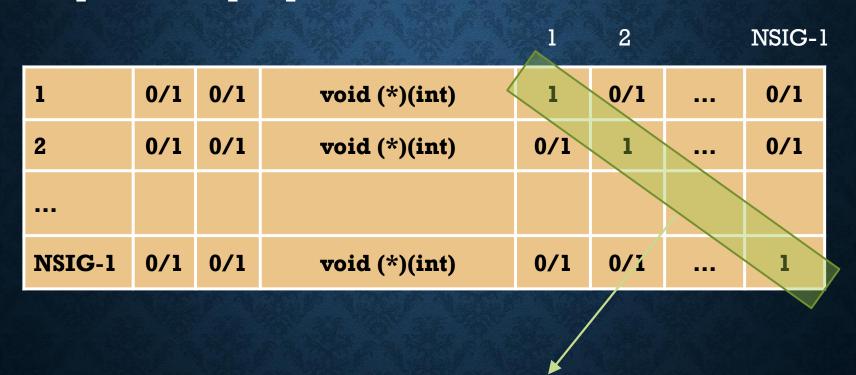
handler (si le signal est armé grâce à *sigaction*)

Table pour chaque processus:

					2		NSIG-1
1	0/1	0/1	void (*)(int)	1	0/1		0/1
2	0/1	0/1	void (*)(int)	0/1	1		0/1
NSIG-1	0/1	0/1	void (*)(int)	0/1	0/1	•••	1

masque temporaire associé au handler (se superpose au masque de processus pendant l'exécution du handler)

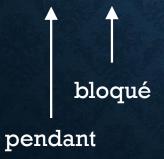
Table pour chaque processus:



un signal est automatiquement bloqué pendant l'exécution de son handler

Table pour chaque processus:

	N. a.			1	2		NSIG-1
1	0/1	0/1	void (*)(int)	1	0/1	•••	0/1
2	0/1	0/1	void (*)(int)	0/1	1		0/1
NSIG-1	0/1	0/1	void (*)(int)	0/1	0/1	•••	1



Quand un signal i est reçu → pendant[i] mis à l

- si bloqué[i]=0 → le signal est traité et pendant[i] est remis à 0
- si bloqué[i]= $l \rightarrow pendant[i] reste à l$
- si un signal i arrive à nouveau \rightarrow il est perdu!

Jeu de signaux

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

- Fonctions permettant respectivement de vider ou remplir un jeu de signaux, supprimer ou ajouter un signal à un jeu de signaux, tester l'appartenance d'un signal à un jeu de signaux.
- Où: ensemble : jeu de signaux
 sig : numéro de signal ∈ [1,NSIG]
- Renvoie 0 si réussi ; -1 si échec (à l'exception de sigismember)

sigprocmask

- Modification du masque de signal courant pour bloquer/débloquer des signaux
- Où: how: modification de l'action du signal (SIG_BLOCK, SIG_UNBLOCK, SIG_SETMASK)

set : nouveau jeu de signaux (bit= $1 \Rightarrow$ signal à modifier)

- oldset : sauvegarde de l'ancien masque
- Renvoie 0 si réussi ; -1 si échec
- Lors d'un *fork*, le processus fils reçoit le même masque de signaux que son père.

Exemple

```
// Définition du masque de signaux à bloquer
sigset t newmask, oldmask;
sigemptyset(&newmask);
sigaddset(&newmask, SIGINT);
sigaddset(&newmask, SIGPIPE);
// Bloquer les signaux SIGINT et SIGPIPE
sigprocmask(SIG BLOCK, &newmask, &oldmask);
/**** CODE CRITIQUE ****/
// Restaurer le masque
sigprocmask(SIG SETMASK, &oldmask, NULL);
```



sigpending

```
#include <signal.h>
int sigpending (sigset_t* set);
```

- Consultation de la liste des signaux bloqués en attente
- Où: set : copie du jeu de signaux en attente (bloqués par le masque de signaux défini par sigprocmask)
- Renvoie 0 si réussi ; -1 si échec

COSTX

sigsuspend

```
#include <signal.h>
int sigsuspend (const sigset t* mask);
```

- Méthode sûre (atomique) pour modifier temporairement le masque des signaux et se mettre en attente (≅ pause() amélioré); elle permet de contrôler le moment de la délivrance d'un signal
- Où: mask: masque remplaçant temporairement le masque de signaux du processus pour permettre la délivrance d'un signal bloqué en le débloquant temporairement (bit=0 ⇒ signal débloqué)
- Si *mask* autorise la délivrance d'un signal détecté comme étant en attente, l'action du signal est immédiatement effectuée et le masque original est restitué; sinon, le programme est suspendu indéfiniment jusqu'à ce qu'un signal non bloqué soit reçu
- Renvoie toujours -1 avec errno = EINTR

Exemple

```
/**** DEBUT DE CODE CRITIQUE ****/
sigset t pending, notpipe;
// Masque où tous les signaux sont activés sauf SIGPIPE
sigfillset(&notpipe);
sigdelset(&notpipe, SIGPIPE);
// Récupération des signaux en attente
sigpending(&pending);
// Vérifier si SIGPIPE est en attente
if (sigismember(&pending, SIGPIPE)) {
   // Autorisation de l'émission de SIGPIPE
   sigsuspend(&notpipe);
  *** FIN DE CODE CRITIQUE ****/
```