

**I2181-B**  
**LINUX :**  
**APPELS SYSTÈME**

**LES SIGNAUX**  
**(SUITE)**

# Structures internes associées aux signaux

## Table pour chaque processus:

1	0/1	0/1	SIG_DFL / SIG_IGN / void (*handler) (int)
2	0/1	0/1	SIG_DFL / SIG_IGN / void (*handler) (int)
3	0/1	0/1	SIG_DFL / SIG_IGN / void (*handler) (int)
...	...	...	...
NSIG-1	0/1	0/1	SIG_DFL / SIG_IGN / void (*handler) (int)




Diagram illustrating the components of signal handling:

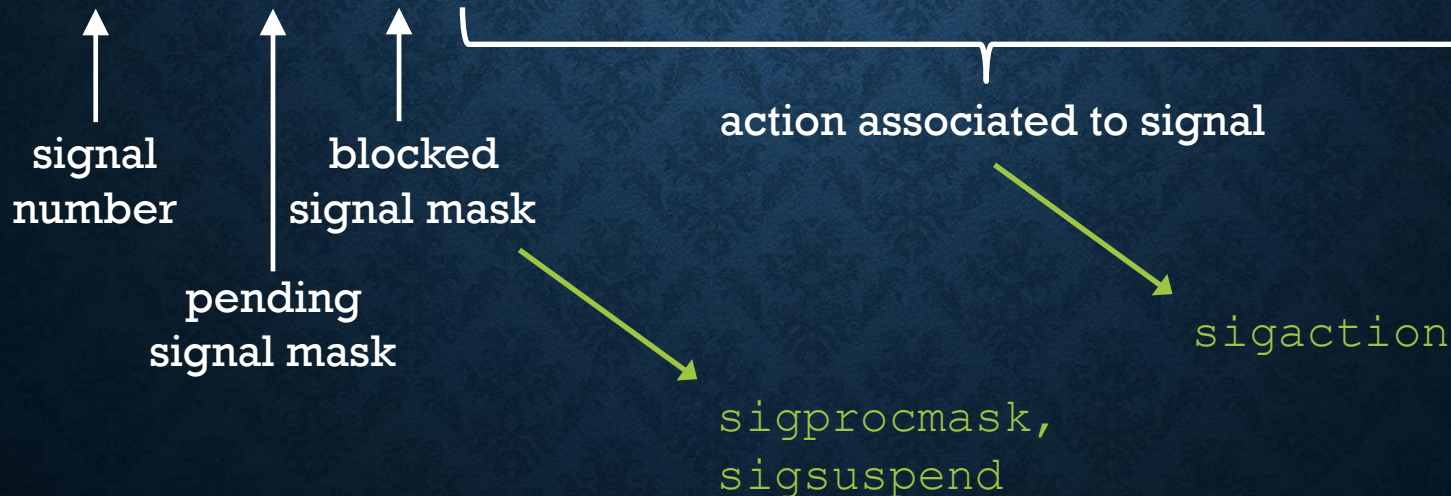
- signal number
- blocked signal mask
- pending signal mask
- action associated to signal



# Structures internes associées aux signaux

Table pour chaque processus:

1	0/1	0/1	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>
2	0/1	0/1	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>
3	0/1	0/1	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>
...	...	...	...
<b>NSIG-1</b>	<b>0/1</b>	<b>0/1</b>	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>



# Structures internes associées aux signaux

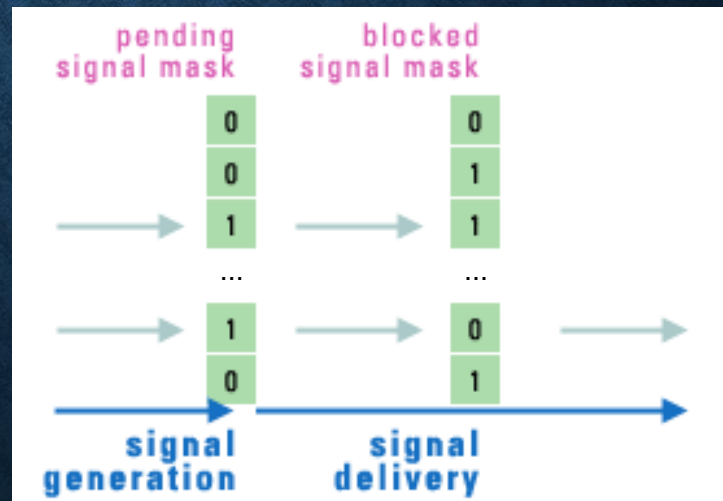
Table pour chaque processus:

1	0/1	0/1	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>
2	0/1	0/1	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>
3	0/1	0/1	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>
...	...	...	...
<b>NSIG-1</b>	<b>0/1</b>	<b>0/1</b>	<b>SIG_DFL / SIG_IGN / void (*handler) (int)</b>

↑  
signal  
number

↑  
pending  
signal mask

↑  
blocked  
signal mask





# Jeu de signaux

```
#include <signal.h>
```

```
int sigemptyset (sigset_t* ensemble);
```

```
int sigaddset (sigset_t* ensemble, int sig);
```

```
int sigdelset (sigset_t* ensemble, int sig);
```

```
int sigfillset (sigset_t* ensemble);
```

```
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  $\in [1, \text{NSIG}]$
  - Renvoie 0 si réussi ; -1 si échec (à l'exception de *sigismember*)
- ➔ Pour manipuler des masques de signaux



# sigprocmask

```
#include <signal.h>
```

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

- Modification du masque de signal courant pour bloquer/débloquer des signaux ; objectif: contrôler quand un signal sera délivré au processus
- Où: **how** : modification de l'action du signal  
(SIG\_BLOCK, SIG\_UNBLOCK, SIG\_SETMASK)  
**set** : nouveau jeu de signaux  
**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.

# sigsuspend

```
#include <signal.h>
```

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

- Modification temporaire du masque de signal courant pour débloquer des signaux  
(cf. exemple14)
- Renvoie toujours -1 (puisque sigsuspend() se termine lorsqu'il est interrompu par un signal)



# alarm

```
#include <unistd.h>
```

```
unsigned int alarm (unsigned int seconds);
```

- Où: *seconds* : durée de temporisation (non bloquante!) 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