UNIVERSITÉ PARIS SUD

IUT D'ORSAY

M3101 · Principes des systèmes d'exploitation

Les signaux

Les signaux

Les signaux sont des interruptions logicielles.

Représenté par un numéro et un nom symbolique (`signal.h`) :

```
SIGKILL    9   kill -9
SIGSEGV   11   segmentation fault
```

Les numéros varient selon l'architecture.

Les signaux peuvent être émis

- par l'utilisateur : `<Ctrl-C>`, `<Ctrl-Z>`, commande `kill`
- par un autre processus : primitives `kill()`, `sigqueue()`
- par le noyau : violation mémoire, instruction invalide, etc.

Traitement des signaux

La réaction par défaut dépend du signal :

SIGHUP, SIGINT, SIGKILL, SIGSEGV, SIGPIPE, SIGALRM, SIGALRM, SIGTERM, SIGUSR1, SIGUSR2	terminaison
SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU	arrêt
SIGCONT	reprise
SIGCHLD	ignoré

Les processus peuvent

- ignorer le signal
- bloquer le signal
- capter et gérer le signal avec une fonction gestionnaire

SIGKILL et SIGSTOP ne peuvent pas être captés, bloqués ou ignorés.

Les informations sur les signaux

Pour chaque type de signal, le PCB d'un processus dit :

- est-ce que le signal est bloqué par le processus ?
- est-ce que le signal est en attente (*pending signal*) ?
- quelle action déclencher en cas de réception ?
  - le traitement par défaut
  - ignorer
  - appeler une fonction gestionnaire

Un processus fils

- hérite les signaux bloqués
- hérite les traitements de signaux
- n'hérite pas les signaux en attente

À l'arrivée d'un signal

Q : Que se passe-t-il quand le signal X arrive ?

R : Si le signal X n'est pas ignoré, alors il est mis en attente.

Q : Que se passe-t-il si le signal X était déjà en attente ?

R : Rien. Le signal X reste en attente, il ne sera traité qu'une fois.

Q : Quand est-ce qu'il sera traité ?

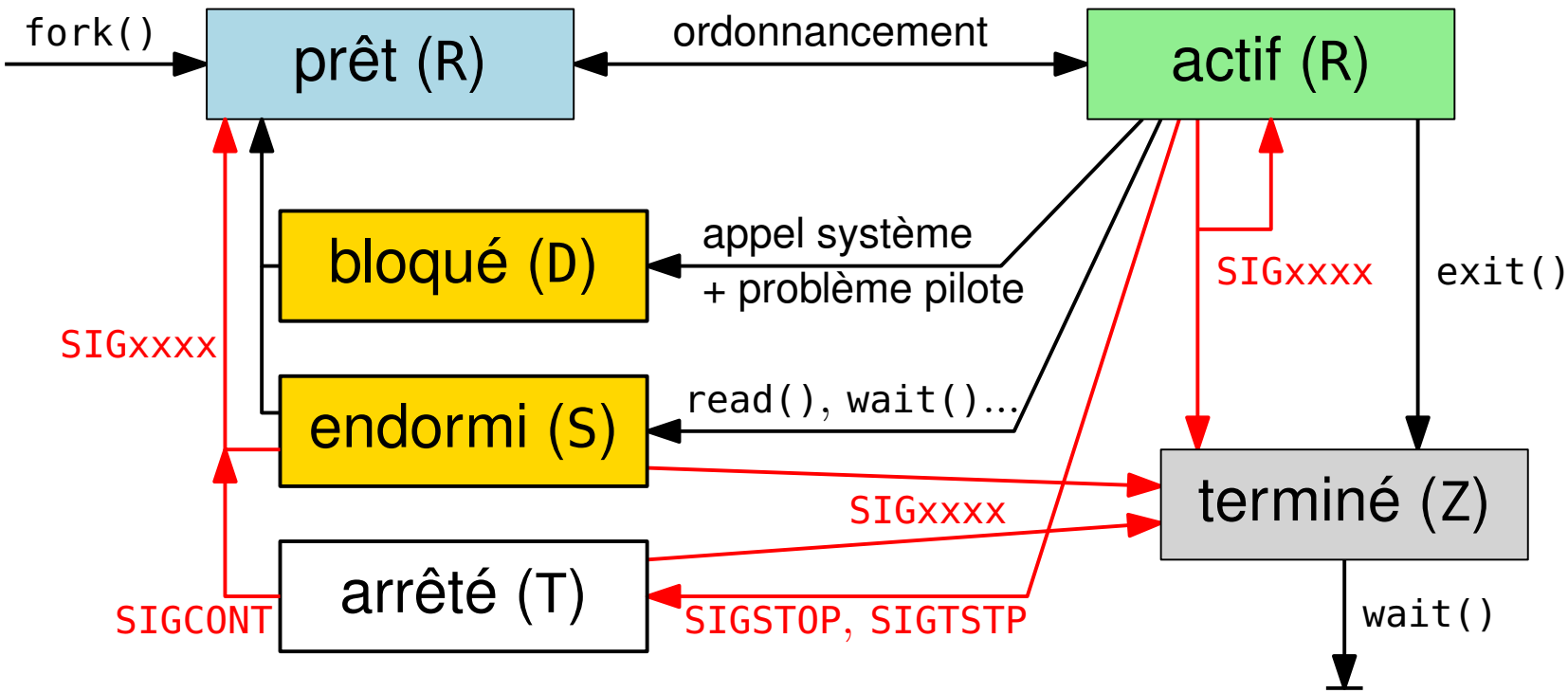
R : Si le signal X n'est pas bloqué, alors il est traité
quand le processus devient actif en mode utilisateur.

Q : Et si le processus est dans un appel d'une primitive système ?

R : Si le processus est endormi dans un appel système (état S),
alors l'appel est interrompu et on revient en mode utilisateur.

Nota : certaines primitives sont automatiquement relancées une fois le signal traité.

La cycle de vie d'un processus



Envoyer un signal

```
int kill(pid_t dest, int nom_ou_numero_du_signal);
```

Pour qui sonne le glas ?

`dest` > 0	$\Rightarrow$	le processus avec le PID indiqué
`dest` $= 0$	$\Rightarrow$	tous les processus du groupe de l'appelant
`dest` $= -1$	$\Rightarrow$	tous les processus de l'utilisateur (sauf `root`)
`dest` < -1	$\Rightarrow$	tous les processus du groupe indiqué

Renvoie -1 en cas d'échec.

Envoyer un signal

```
unsigned int alarm(unsigned int nbSec);
```

Un signal SIGALRM sera envoyé à l'appelant dans nbSec secondes.

Annule et remplace toute alarme précédente.
$\Rightarrow$ Renvoie le nombre de secondes qu'il lui restait.

Renvoie -1 en cas d'échec.

Ne pas utiliser en même temps que `sleep()`.

Gérer un signal

```
sighandler_t signal(int sig, sighandler_t gestionnaire)
```

Pour restaurer le traitement par défaut :

```
signal(SIGxxxx, SIG_DFL);
```

Pour ignorer le signal :

```
signal(SIGxxxx, SIG_IGN);
```

Pour installer une fonction gestionnaire :

```
void traiter_SIGxxxx(int sig) { ... }
...
signal(SIGxxxx, traiter_SIGxxxx);
```

Renvoie la valeur précédente du gestionnaire (ou `SIG_ERR`)

Gérer un signal

La sémantique de `signal()` varie dans les différents systèmes Unix.

Le standard POSIX introduit une primitive puissante et portable :

```
int sigaction(                    int sig,
              const struct sigaction *nouvelleAction,
                    struct sigaction *ancienneAction)
```

Renvoie 0 en cas de succès et -1 en cas d'erreur.

Sauvegarde la valeur précédente de l'action dans `*ancienneAction`.

Sous Linux, `signal(SIGxxxx, gestionnaire);` est équivalent à

```
struct sigaction action = {0};
action.sa_handler = gestionnaire;
action.sa_flags = SA_RESTART;
sigaction(SIGxxxx, &action, NULL);
```

Avez-vous dit quelque chose ?

```c
int counter = 0;
void handler(int signum) {
  if (++counter == 3) exit(0); }

int main() {
  signal(SIGINT, handler); // SIGINT est émis par <Ctrl-C>
  while (1) { pause(); /* attend un signal quelconque */ }
  return 0; }
```

Dans un terminal :

```
> ./sigint
^C^C^C
>
```

La masque de signaux

Pour bloquer ou débloquer des signaux :

```
int sigprocmask(            int mode,
                const sigset_t *nouvelleMasque,
                      sigset_t *ancienneMasque)
```

`mode`	`SIG_BLOCK`, `SIG_UNBLOCK` ou `SIG_SETMASK`
`sigset_t`	les ensembles de signaux, voir `man sigsetops`

À la réception d'un signal bloqué et non-ignoré,

- le signal est mis en attente (*pending signal*)
- le traitement est reportée jusqu'à ce que le signal soit débloqué

Par défaut, tout signal est bloqué pendant l'exécution de son gestionnaire.