Signaux

Abdelkader Gouaïch

IUT de Montpellier

2014-2015

Définition d'un signal

Définition

C'est un événement (i) généré par le système Linux en réponse à des conditions particulières qui (ii) provoquera une réaction d'un processus à sa réception.

Trois phases :

1. Génération
2. Réception
3. Traitement

Définition d'un signal

- On utilise souvent les termes :
  - Déclencher/Lever : pour parler de la génération d'un signal
  - Capturer/Gérer : pour parler du traitement par un processus d'un signal

Conditions pour lever des signaux

- Exemples de situations pour lever des signaux :
- Erreurs : violation des règles des segments mémoires, instructions illégales, division par zéro
- Envoyés par le Shell : Comme le Ctrl^C, Ctrl^Z
- Envoyés explicitement par un processus : en utilisant une API

Librairie des signaux

- Tous les noms de signaux commencent avec `SIG`
- Les noms de signaux sont définis dans le fichier entête `signal.h`

Librairie des signaux

- SIGABORT abandon du processus
- SIGALRM alarme horloge
- SIGFPE virgule flottante
- SIGHUP raccrocher/fin de connexion
- SIGILL instruction illégale
- SIGINT interruption du terminal
- SIGKILL tuer le processus
- SIGPIPE écriture dans un tube sans lecteur
- SIGQUIT le terminal quitte
- SIGSEGV accès invalide au segment mémoire
- SIGTERM terminaison
- SIGUSR1 signal laissé pour l'utilisateur
- SIGUSR2 signal laissé pour l'utilisateur

Librairie des signaux

- SIGCHLD le processus fils a terminé ou stoppé son exécution.
- SIGCONT continuer l'exécution.
- SIGSTOP stopper l'exécution
- SIGTSTP un signal stop du terminal

Par le shell

- Le terminal du Shell est déjà configuré pour envoyer des signaux aux processus s'exécutant en avant plan
- Par exemple, Ctrl^C, provoque l'envoi par le Shell du signal SIGINT au processus s'exécutant en avant plan
- Si ce processus ne prévoit pas de capturer le signal SIGINT, alors il s'arrête

La commande `kill`

- On peut également envoyer un signal à un processus particulier en indiquant son PID
- C'est le rôle de la commande `kill`
- `kill -HUP 3422`

- On peut capturer un signal en utilisant la librairie `signal.h`

```
#include <signal.h>
void (*signal(int sig, void (*func)(int)  ) ) (int);
```

- Signal est une fonction qui prend deux paramètres :
  - sig : un entier qui identifie le signal (numéro du signal)
  - fun : est une fonction qui prend en paramètre un entier et ne renvoie aucun résultat
- la fonction fun sera appelée avec comme paramètre le signal reçu.

- Il existe des valeurs spéciales pour le paramètre fonction (fun) :
  - SIG_IGN : ignorer le signal
  - SIG_DFL : remettre le comportement par défaut

Exemple :

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

void ouch(int sig)
{
printf("OH! -J'ai capture le signal %d\n", sig);
(void) signal(SIGINT, SIG_DFL);
}


int main()
{
(void) signal(SIGINT, ouch);
```

```
Quand on fait Ctrl^C
$./testsignal
Bonjour!
Bonjour!
Bonjour!
^C
OH! -J'ai capture le signal 2
Bonjour!
^C
$
```

- Dans un programme on peut envoyer un signal en utilisant la fonction kill

```
#include <sys/types.h>
#include <signal.h>
int kill(pid_t pid, int sig);
```

SIGALARM

- Un programme peut également s'envoyer un signal SIGALARM avec la fonction alarm

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

  - Remarques :
    - On processus ne peut charger qu'une seule alarme
    - 0 comme paramètre réinitialise l'alarme
    - Rappeler alarm avant la fin de la première alarme retourne le temps restant