IUT de Montpellier - Programmation Systeme
Les signaux

29 septembre 2022

1 Manipulation des signaux en Shell (rappels)

Connectez vous sous Linux et ouvrez un terminal.

1.1 Envoi de signal avec la commande kill

- Lancer la commande sleep pour 100 secondes.
- Quel est le PID de votre `sleep` ?
- ($\star$) Combiner à l'aide de pipe les commandes `ps`, `grep` et `cut` pour trouver le PID du sleep . cette Lister les processus en cours dans votre terminal. Quel est le PID de votre `sleep` ?
  Indice on peut utiliser l'option `-v` de `grep` pour exclure les lignes qui contiennent une expression.
- A l'aide de la commande `kill` envoyer le signal 9 au processus `sleep`. Que se passe t"il ?

1.2 Signaux et gestion des droits

- A l'aide de la commande ps -Al trouver le PID et UID du processus *netns*
- A l'aide du fichier */etc/passwd*, retrouver à qui correspond cet UID ?
- Envoyer le signal -9 à ce processus. Que se passe t'il ?

1.3 Signaux et hiérarchie de processus

- Notez le PID du bash de votre terminal.
- Lancer la commande sleep 200 en arrière plan.
- Quel est le PPID du processus sleep ?
- Ouvrez un second terminal. Et depuis ce terminal envoyer le signal 9 au PID du bash du premier terminal.
- Qu'est il arrivé au processus sleep ?

Sauvegardez votre travail, rechercher le PID du processus `systemd` dont l'UID est le votre à l'aide de la commande $ps - Al$ puis envoyez lui le signal 9.

2 Programmation C

2.1 Le programme myKill

A l'aide de la fonction C *kill*, écrivez le programme myKill qui prend exactement 2 arguments :

1. le PID du processus à qui envoyer le signal
2. le numéro du signal à envoyer

```
1 #include <signal.h>
2 int  kill(pid_t pid, int sig);
```

Testez l'exécutable *myKill* sur un processus *sleep* (comme dans la première partie du TP). Attention l'ordre des arguments n'est pas le même que *kill* et le numéro du signal n'est pas précédé de $-$.

2.2 Capturer les signaux

Le petit programme suivant illustre comment capturer un signal en C. L'appel $signal(n, f)$ permet de déclencher la fonction f lorsque le signal numéro n est reçu.

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <stdbool.h>
#include <unistd.h>

void sigUp( int code ) {
    printf( ">>> SIGUP received [%d]\n", code );
}
void sigInt( int code ) {
    fprintf( stderr, ">>> SIGINT received [%d]\n", code );
}

int main (int argc, char *argv[]) {
    // On enregistre les gestionnaires de signaux.
    signal( SIGHUP, &sigUp );
    signal( SIGINT, &sigInt);

    while( 30 ) {
        printf("I am still alive\n");
        sleep(1);
    }
}
```

A l'aide de la page man de signal trouver les numéros de signaux qui se cachent derrière $SIGHUP$ et $SIGINT$. Compilez et lancer le programme ci-dessus et envoyer lui des signaux avec *kill* ou *myKill*

3 Projet : Kenny

3.1 Ecrire le programme kenny , qui intercepte les signaux.

- Le programme Kenny prend un seul paramètre qui initialise son nombre de points de vie.
- Toutes les 3 secondes, il affiche son nombre de point de vie puis la phrase "c est trop injuste".avant de se rendormir pour 3 secondes.
- kenny intercepte tous les signaux de 1 à 9. A chaque fois qu'il reçoit un signal il affiche le message « c'est injuste » et décrément ses points de vie en fonction du numéro du signal.
- Quand il n'a plus de points de vie, bah il meurt.

3.2 Jouons un peu avec Kenny :

- Lancer un Kenny avec 20 points de vie.
- Attaquer le avec la commande Linux kill et votre programme myKill jusqu'à ce que mort s'en suive.
  En pro du bash, vous pouvez
  1. récupérer les PID des programmes nommés "kenny" : `ps |grep kenny | cut -d " " -f1`
  2. envoyer direct un signal à ces programmes `kill -2 ‘ps |grep kenny | cut -d " " -f1‘`
- Lancer un Kenny avec 100 points de vie.
- Attaquez le une seule fois avec le signal 9, que se passe t'il ?

3.3 Rendre Kenny invicible

- Ajouter la méthode invicible à votre programme.

```c
void invicible(int sig) {
    int delay = 10;
    while (delay--> 0) {
        printf("je suis invincible\n");
```

```
5           sleep(1);
6       }
7   }
```

— à l'aide de signal, abonner Kenny à la fonction invicible, lorsqu'il reçoit le signal 10.
— envoyer le signal 10 à Kenny, puis plusieurs signaux 1. Que se passe t'il ?
— "protéger" l'execution de la fonction invincible d'une interuption par un autre signal

1. déclarer un objet de type `sigset_t` et remplisser le à l'aide des fonctions `sigemptyset,sigfillset,sigaddset et sigdelset`.
2. utiliser l'appel `sigprocmask` pour masquer les signaux durant au début de la fonction invincible, puis retirer ce masque en fin de fonction.