Les Signaux En Linux

Realise par :

Youssef Ait Ouchaour

GI1 , N° 4

Encadre par :

Pr. Sara Baghdadi

Plan

1. Definition
2. Les Signaux
3. Principe de Fonctionnement
4. implementation

1. Definition

- Sur un système Unix, il est fréquent que des processus différents doivent communiquer entre eux et coordonner leurs activités.
- Signal = un mécanisme de communication inter-processus, est un événement asynchrone qui est livré à un processus.…
- Asynchrone signifie que l'événement peut survenir à tout moment peut être sans rapport avec l'exécution du processus.

1. Definition

- Souvent utilisé pour certaines conditions d'erreur telles que:

- Prévenir un processus qu'il effectue une opération invalide ou instructions illégales(accès mémoire invalide, division par zéro…).

- Notifier un processus lorsque sa configuration change.

- Et pour arrêter un processus (par tape ctrl-C).

2. Les Signaux (31 in POSIX)

- voici quelques signaux prédéfinis dans les systèmes d'unix:

- SIGHUP : fermeture terminal à tous les processus attachés.
- SIGKILL : pour terminer un processus .
- SIGSEGV : accès mémoire invalide (envoyé par le noyau).
- SIGUSR1 : libre, sémantique définie pour chaque processus.
- SIGUSR2 : libre, sémantique définie pour chaque processus.
- SIGSTOP: Stop a process execution

2. Les Signaux (31 in POSIX)

- SIGINT: Interruption du processus. Peut être généré par ’Delete’ ou bien ‘ctrl_C’ keys.
- SIGTSTP : demande au système de suspendre un processus (généré par control-z)
- SIGCONT : demande au système de le redémarrer (avec bg ou fg)…

Fonctions et cmnds pour signaux:

kill()

- Un processus peut envoyer un signal à un destinataire avec :

  kill -sig pid

  • sig : numéro de signal (nombre ou symbole comme USR1).

  • pid : PID du processus destinataire.

- Par exemple :

```
$ kill -KILL 4481
```

- Return 0 if ok, -1 on error.

pause()

- Suspendre le processus d'appel jusqu'à ce qu'un signal soit capté.

- ```
  #include <unistd.h>
  int pause(void);
  ```

usleep(), sleep()

- fait dormir le processus appelant pendant un nombre de secondes spécifié.
- Exemple : sleep(1) /sleep for 1 minute.
  ```

signal(): library call

- Les programmes peuvent gérer les signaux à l'aide de la bibliothèque de fcts des signaux.
-
```
#include <signal.h>
typedef void Sigfunc(int);  /* my defn */

Sigfunc *signal( int signo, Sigfunc *handler );
```

  - signo est le pid du signal à gérer
  - sgfunc *handler définit comment gérer le signal(hundler)
  - Retourne la disposition précédente si ok, ou SIG_ERR en cas d'erreur.

sigaction()

- Remplace (plus puissant que signal()), sigaction() peut être utilisé pour coder un signal non réinitialisable

- ```
  #include <signal.h>

  int sigaction(int sig, const struct
  sigaction *act, struct sigaction
  *oldact );
  ```
  ```

sigaction Structure

- La fonction sigaction définit l'action associée au signal sig .

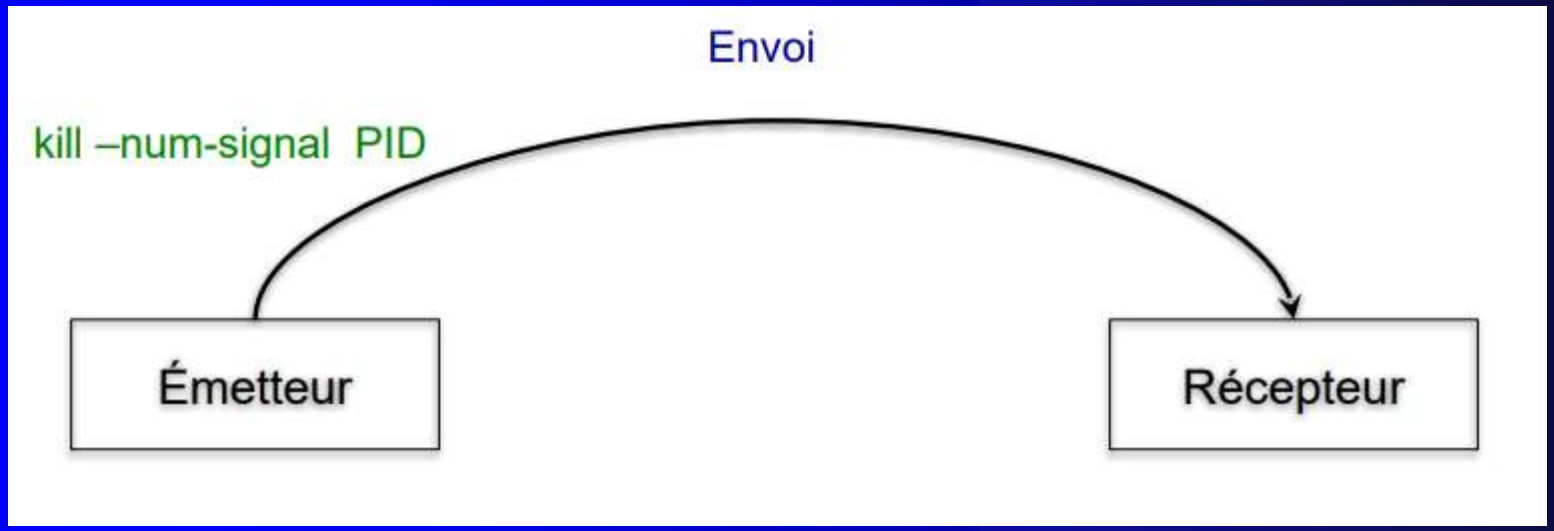
Si oact n'est pas nul, sigaction écrit l'action de signal précédente dans le lieu auquel il se réfère. Si l'acte n'est pasnull, l'action pour

le signal spécifié est définie.

```
struct sigaction
    {
    void       (*sa_handler)( int );
                  /* action to be taken or SIG_IGN, SIG_DFL */
    sigset_t   sa_mask; /* additional signal to be blocked */
    int        sa_flags; /* modifies action of the signal */
    void       (*sa_sigaction)( int, siginfo_t *, void * );

    }
```

3. Principe de Fonctionnement



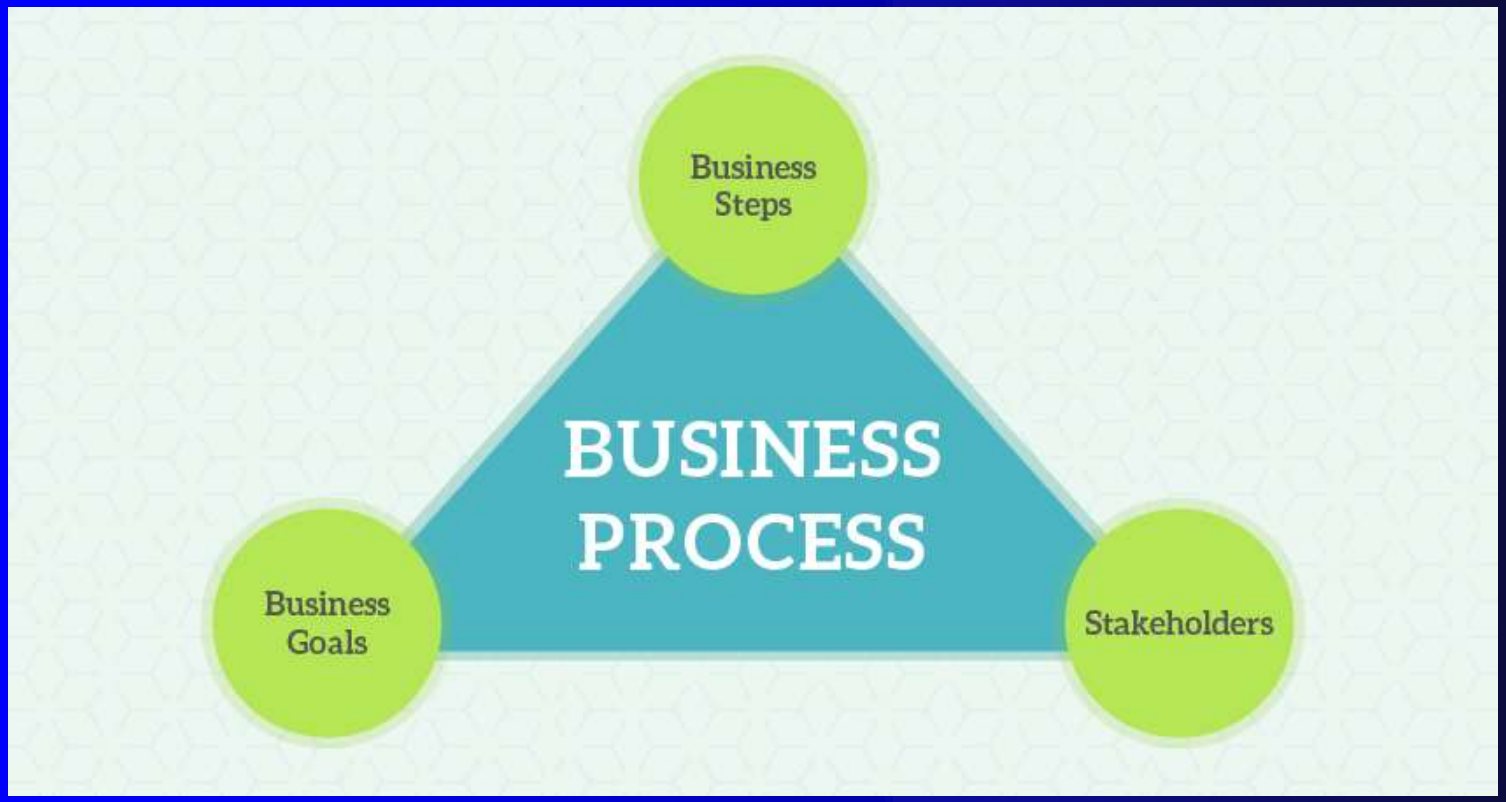
- Émetteur envoie un message à un processus

• Un message est limité à un nombre compris entre 1 et 31 .

• Tout signal émis est livré (sauf si le même numéro de signal est émis une seconde fois avant réception – dans ce cas le deuxième signal est perdu) .

• Ordre de réception aléatoire.

4.implementation(client-serveur)



• le code suivant est codee par Language C et Linux fonctions.