

Systeme UNIX et programmation

Dr. Davy MOUSSAVOU- Consultant Linux, Certifié
Linux International

Chapitre 6: Les signaux

Contenu du cours

Séquence 1: Concepts théoriques

Séquence 2: Envoi d'un signal

Séquence 3: Réception d'un signal

Séquence 1 : Concepts théoriques

Les signaux sont utilisés ?

- Les signaux sont utilisés pour la communication interprocessus.
- C'est un moyen pour le noyau d'envoyer des informations à un processus.
- Certains signaux traduisent des événements détectés par le noyau.

Comment fonctionnent les signaux?

- Une liste de signaux est mis à la disposition de chaque processus;
- Chaque processus associe une fonction de traitement à chaque signaux(ex: fermer, réduire ou agrandir la fenêtre);
- Ainsi un processus peut recevoir un signal à tout moment.

Qu'est qui caractérise les signaux

- Chaque signal est identifié par son nom;
- Un nom de signal est associé à un numéro;
- Chaque signal est associé à un traitement par défaut;
- Un signal peut être ignoré.

Séquence 1: Concepts théoriques	Définition
Séquence 2: Envoie d'un signal	Principe
Séquence 3: Réception d'un signal	Caractéristiques
TP	Exemple de signaux

Description de quelques signaux

- ❑ **SIGABRT:** Il est déclenché lors d'appel de la fonction **abort()**. Celle-ci indique la fin anormale d'un processus. Cas d'une erreur de programmation.
- ❑ **SIGKILL:** Ce signal déclenche l'arrêt immédiat d'un processus qui le reçoit. Il faut souligner que le processus **init** ne peut pas recevoir ce signal. Avec ce signal, le programmeur a la garantie de reprendre la main sur un programme donné.
- ❑ **SIGQUIT:** Ce signal est envoyé lors de la frappe de la touche QUIT(Contrôle-\\). Cette touche correspond aux touches Ctrl-AltGr-\\ sur le clavier azerty. Ce signal a pour effet, la fin d'un processus qui ne l'ignore pas.
- ❑ **SIGSTOP:** Un processus qui reçoit ce signal est stoppé.

Description de quelques signaux

❑ **SIGCONT:** Ce signal a pour effet, de relancer un processus stoppé.

❑ **SIGTSTP:** Il stop un processus en avant plan. Il est déclenché à la réception des touches Ctrl-Z.

❑ **SIGTERM:** Il a pour effet de terminer « *gentiment* » un processus. C'est le signal est envoyé par défaut par la commande **kill** d'UNIX.

❑ **SIGTRAP:** Ce signal est déclenché lorsqu'un processus a atteint son point d'arrêt. Ce signal est utilisé par des débogueurs.

Description de quelques signaux

❑ **SIGUSER1 et SIGUSER2:** Ces signaux ont pour effet de terminer un programme.

❑ **SIGWINCH:** Ce signal a pour effet d'indiquer que la taille du terminal a été modifié.

❑ **SIGXCPU et SIGXFSZ:** Ces signaux sont émis par le noyau lorsqu'un processus dépasse une de ses limites souples des ressources système (temps cpu, taille maxi d'un fichier ouvert, tentative de création d'un fichier trop grand, etc.)

Séquence 2 : Envoi d'un signal

La fonction `kill()`

L'émission d'un signal se fait avec l'appel système `kill()`.

\$ man 2 kill

Prototype:

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

```
int kill(pid_t pid, int numeor_signal);
```

Exemple:

```
kill(getpid, TERM);
```

```
Kill(getpid,5);
```

Séquence 3 : Réception d'un signal

Que passe-t-il à la Réception d'un signal ?

- A la réception d'un signal, un processus peut installer un handler, c'est-à-dire une routine spécifique sera invoquée à la réception du signal.
- A la réception d'un signal les actions suivantes peuvent être effectuées:
 - ☐ Ignorer le signal. Ce comportement est représenté par la constante symbolique **SIG_IGN**.
 - ☐ Laisser le noyau appliquer le comportement par défaut. Dans ce cas de figure, la constante symbolique **SIG_DFL** est utilisée.
- Pour indiquer ces actions aux noyaux, les appels systèmes `signal()` et `sigaction` sont utilisés

L'appel system `signal()`

➤ `signal()`

✓ Avantages

- Son implémentation est simple.

✓ Inconvénients

- Problème de fiabilité à la délivrance des signaux;
- Problème de compatibilité avec certains drivers des systèmes Unix.

L'appel system **signal()**

- La valeur de retour, il renvoie un pointeur sur l'ancien handler.
- Ce qui permet de garder un image pour le réutiliser.

Prototype:

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

```
void (*signal(int sig, void (*handler)()));()
```


L'appel system **signal()**

Exemple d'usage de l'appel système `signal()`.

```
signal(SIGTERM,SIG_DFL);
```

```
signal(SIGCHLD, SIG_IGN);
```

L'appel system **sigaction()**

➤ sigaction()

✓ avantages

- Précision dans la définition du comportement souhaité par le programmeur

✓ Inconvénients

- Problème de complexité du code due au remplissage d'une structure
- Il n'est pas utilisable sur tous les systèmes de d'exploitation.

Séquence 1: Concepts théoriques
Séquence 2: Envoie d'un signal
Séquence 3: Réception d'un signal
TP

