LINUX : Gestion des signaux

Table of Contents

Il est possible de modifier le comportement des signaux envoyés au shell en utilisant la commande trap.

Principaux signaux

Liste des signaux :

```
$ kill -l
```

```
  1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
  6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
  11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15)
SIGTERM
  16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20)
SIGTSTP
  21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25)
SIGXFSZ
  26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
```

```
  31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37)
SIGRTMIN+3
  38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42)
SIGRTMIN+8
  43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47)
SIGRTMIN+13
  48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52)
SIGRTMAX-12
  53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57)
SIGRTMAX-7
  58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62)
SIGRTMAX-2
  63) SIGRTMAX-1  64) SIGRTMAX
```

Détail des signaux :

```
HUP    hangup : envoie un signal de réinitialisation
INT    Interruption
QUIT    Core dump
ILL    Le programme tente d'exécuter du code malformé, inconnu ou avec de
mauvais privilèges
TRAP    Le programme envoie un signal au débugger (message capturé)
IOT    idem SIGABRT : interruption du programme
EMT    emulator trap : un programme émulé ou virtualisé a posé problème
FPE    floating-point exception : le programme a réalisé une opération
arithmétique erronée
KILL    arrête le programme immédiatement
BUS    Le programme a causé une erreur de bus
SEGV    Segmentation fault : le programme fait référence à un mauvais
emplacement de mémoire
SYS    Un mauvais argument est passé en paramètre
PIPE    Un programme tente d'écrire dans un pipe sans processus connecté à
l'autre bout
ALRM    La limite de temps est dépassée
TERM    Envoie un signal au programme pour le terminer
USR1/USR2    Envoie un signal dans des conditions définies par un
utilisateur
CHLD/CLD    child : signal envoyé par un programme lorsqu'un processus fils
est achevé
PWR    power : le système subit un problème d'alimentation
VTALRM    virtual alarm : signal envoyé lorsque le temps limite a été
dépassé
PROF    profiler : signal envoyé lorsqu'un timer a expiré
POLL    polling : un problème est survenu lors d'un événement I/O asynchrone
WINCH    window [size] change : signal envoyé au programme lorsque la
fenêtre de contrôle change de taille
STOP    signal demandant au programme de se suspendre
TSTP    tty stop : signal envoyé au programme lorsqu'un terminal suspend ses
requêtes
```

```
CONT    Redémarre un programme suspendu par STOP
TTIN    Le programme tente de lire tty alors qu'il est en arrière-plan
TTOU    Le programme tente d'écrire sur tty alors qu'il est en arrière-plan
URG    Un socket a une donnée urgente à lire
LOST    Le verrou d'un fichier a été perdu
XCPU    Le programme a utilisé le CPU prend trop longtemps
XFSZ    Le fichier a dépassé la taille maximale autorisée
RTMIN/RTMIN+n    real-time minimum : signaux définis par l'application
RTMAX/RTMAX-n    real-time maximum : signaux définis par l'application
```

Utilistaion

Dans les commandes, les signaux peuvent être exprimés sous forme numérique ou symbolique. Les signaux HUP, INT, TERM et KILL possèdent la même valeur numérique sur toutes les plates-formes Unix, ce qui n'est pas le cas de tous les signaux. Il est donc préférable d'utiliser la forme symbolique.

Le signal INT

Est généré à partir du clavier.Il est utilisé pour tuer le processus qui tourne en avant plan. Pour connaitre la saisie clavier correspondant au signal INT, voir le paramètre intr de la commande stty -a.

```
$ stty -a
```

```
  speed 38400 baud; rows 36; columns 134; line = 0;
  intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>; eol2
= <undef>; swtch = <undef>; start = ^Q; stop = ^S;
  susp = ^Z; rprnt = ^R; werase = ^W; lnext = ^V; flush = ^O; min = 1; time
= 0;
  -parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
  -ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -
ixoff -iuclc -ixany -imaxbel -iutf8
  opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0
vt0 ff0
  isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -
echoprt echoctl echoke
```

Ignorer un signal

Syntaxe :

```
trap ' ' sig1 sig2
```

Exemple :

```
# Le shell courant correspond au PID 30819

$ echo $$

30819

# Modification des signaux HUP et TERM

$ trap '' HUP TERM

# Envoi des signaux HUP et TERM

$ kill -HUP 30819
$ kill -TERM 30819

# Les signaux sont ignorés et le processus est toujours actif
$ echo $$

30819
```

Modifier le traitement associé à un signal

Syntaxe :

```
trap 'cmd1 ; cmd2 ; cmd3 ; ..... ; cmdn' sig1 sig2
```

Exemple :

```
# Le shell courant correspond au PID 23217

$ ps
  PID TTY          TIME CMD
22109 pts/0    00:00:00 bash
23217 pts/0    00:00:00 bash
23465 pts/0    00:00:00 ps

$ echo $$

23217

# Modification du traitement associé au signal ^C (ctrl+c)
# On demande au shell d'afficher le message "Signal INT reçu" après avoir
appuyer sur ^C (ctrl+c)
```

```
$ trap 'echo "Signal INT reçu" ; exit 1' INT
$ ^C     # Saisie
$ Signal INT reçu

# Le shell courant correspondant au PID 23217 n'existe plus

$ ps
  PID TTY          TIME CMD
22109 pts/0    00:00:00 bash
24229 pts/0    00:00:00 ps

$ echo $$

22109
```

Repositionner le traitement par défaut du shell vis-à-vis d'un signal

Syntaxe :

```
trap - sig1 sig2 ..... sign
```

Exemple :

```
# Création du fichier /tmp/fichier

$ > /tmp/fichier
$ ls /tmp/fichier
/tmp/fichier

# Modification du traitement associé au signal INT et TERM
# On demande au shell de supprimer le fichier "/tmp/fichier" après avoir
appuyer sur ^C (ctrl+c)

$ trap 'rm -f /tmp/fichier' INT TERM

# Le shell courant correspond au PID 22109

$ echo $$

22109

# Envoi du signal INT "^C" (ctrl+c)

$ ^C

# Le fichier a bien été suprimé
```

```
$ ls /tmp/fichier

ls: impossible d'accéder à /tmp/fichier: Aucun fichier ou dossier de ce type

# Création d'un nouveau fichier /tmp/fichier

$ > /tmp/fichier
$ ls /tmp/fichier
/tmp/fichier

# Traitement associé au signal INT et TERM remis par défaut

$ trap - INT TERM

# Le shell courant correspond au PID 22109

$ echo $$

22109

# Envoi du signal INT "^C" (ctrl+c)

$ ^C

# Le fichier n'a pas été supprimé

$ ls /tmp/fichier
/tmp/fichier
```

Utiliser trap à partir d'un script shell

L'utilisation de trap dans un script shell va permettre de gérer des actions en fonctions de différents signaux reçus.

Exemple :

Dans le script suivant, à la réception d'un signal HUP INT ou TERM, la fonction “fin” est appelée et le fichier $fileTmp est supprimé.

```
$ nl signaux.sh

 1  #!/bin/bash
 2
 3  # Nom du fichier temporaire
 4  fileTmp=/tmp/fileTemp
 5
```

```
 6  # Fonction appelée lors de la réception d'un signal HUP INT TERM
 7  function fin {
 8          echo -e "\nSuppression du fichier $fileTmp"
 9          echo "Fin du script"
10          rm -f $fileTmp
11          ls $fileTmp
12          exit 1
13  }
14
15  # Paramétrage de la fonction "fin" à la réception d'un signal HUP INT
TERM
16  trap fin HUP INT TERM
17
18  # Création du fichier temporaire
19  > $fileTmp
20
21  echo "Lancement du script"
22  # Vérification de la création du fichier temporaire
23  ls $fileTmp
24  sleep 100
25  echo "Arrêt du script"
26  exit 0
```