Cheatography

Signaux Linux Cheat Sheet

by TME520 (TME520) via cheatography.com/20978/cs/3863/

3 règles

1. Seul *root* peut manipuler les processus des autres utilisateurs et du système,
2. Un utilisateur ne peut agir que sur les processus qu'il possède,
3. *init* (PID 1) est parfaitement insensible à tout ça.

Un programme recevant un signal a 4 alternatives

Ignorer	Le signal lui passe dessus comme un pet de coucou sur une toile cirée (c'est le cas quand le logiciel ne dispose d'aucun mécanisme approprié pour réagir ; *blame the developer !*)
Suspendre	L'exécution du processus est suspendue. Il reste en mémoire, attendant une action de son processus père, qui peut le stopper totalement ou poursuivre son déroulement sous d'autres conditions (en tâche de fond, par exemple).
Quitter	Le signal provoque l'arrêt du processus. Dans certains cas, un fichier *core* (comme dans *core dump*) destiné à être exploité par un debugger est écrit sur un disque.
Exécuter	Une fonction dédiée à la gestion du signal est exécutée, puis le logiciel reprend son cours normal.

2 types de signaux

Type	Numéros
Classique	1-31
Temps-réel	32-64

Les signaux « temps-réel »

Tout ce qui est au-delà de 31 correspond à des signaux dits « temps-réel » configurables comme on le veut. Leur nombre exact est défini par les macros SIGRTMIN et SIGRTMAX. Qu'est-ce qui différencie les signaux classiques des signaux TR (*RT* pour *Real Time* en Anglais) ?
Premièrement, au contraire de ce qui se passe pour un signal classique, aucune occurrence d'un signal TR ne se perd, même si elle vient en doublon ou plus. Toutes sont placées dans une file d'attente selon leur ordre de priorité (plus bas est le numéro du signal, plus haute est sa priorité).
Deuxièmement, il est possible de communiquer une valeur entière ou un pointeur en accompagnement d'un signal TR.
Troisièmement, si plusieurs signaux TR du même type sont envoyés à un processus, ils sont exécutés dans leur ordre d'arrivée.
Quatrièmement, les signaux TR ne sont jamais déclenchés par le kernel, mais par l'utilisateur. Ils sont, un peu à la manière de *SIGUSR1* et *SIGUSR2*, définis en fonction des besoins.

Les signaux classiques

1	SIGHUP	Fin de connexion (*hang-up*) du terminal de contrôle. Peut être envoyé manuellement à un daemon afin qu'il se réinitialise.
2	SIGINT	Interruption en provenance du clavier (Ctrl + C). Peut être ignoré, intercepté ou bloqué. Quitte proprement.
3	SIGQUIT	Interruption en provenance du clavier (Ctrl + \). Même chose que SIGINT à cela près qu'un fichier core dump est généré.
4	SIGILL	Le programme a tenté d'exécuter une instruction invalide ou non-supportée par le microprocesseur / le coprocesseur arithmétique.
5	SIGTRAP	Utilisé pour indiquer à un programme en cours de *debug* qu'un point d'arrêt a été atteint.
6	SIGABRT	Ce signal est émis lorsqu'un programme rencontre un problème forçant son arrêt. Génère un *core dump*.

Les signaux classiques (cont)		
7	SIGBUS	Ce signal est émis lorsqu'un programme tente d'utiliser une adresse mémoire invalide (mauvais alignement, adresse impaire, adresse indisponible…).
8	SIGFPE	Erreur arithmétique (division par zéro). Génère un *core dump* et tombe raide mort.
9	SIGKILL	On signale au kernel qu'on veut stopper net un processus. Ne peut être ni ignoré, ni intercepté, ni bloqué.
10	SIGUSR1	Par défaut un logiciel recevant ce signal s'arrête, à moins que son programmeur n'ait prévu quelque chose d'autre (afficher des statistiques, écrire dans un fichier journal…). C'est en quelque sorte un signal personnalisable pouvant servir à déclencher ce qu'on veut.
11	SIGSEGV	Erreur de protection mémoire (*segmentation fault*). Cela se produit quand un processus demande à écrire à une adresse ne figurant pas dans son espace d'adressage.
12	SIGUSR2	Idem SIGUSR1.
13	SIGPIPE	Le fameux *Broken pipe*. Survient quand on envoie des données dans un tuyau (*pipe*) alors que personne n'est à l'autre bout pour les récupérer.
14	SIGALRM	Utilisé pour programmer des temporisations sur des opérations bloquantes ou susceptibles de l'être (lecture/écriture sur un volume réseau par exemple).
15	SIGTERM	Une invitation à quitter proprement est envoyée au processus. Peut être ignoré, intercepté ou bloqué. La différence avec SIGINT ? Aucune combinaison de touches ne permet de générer un SIGTERM.
16	SIGSTKFLT	Indique une erreur de pile. Normalement pas utilisé. Par défaut, provoque l'arrêt du programme.
17	SIGCHLD	Un des processus fils est mort ou a été arrêté.
18	SIGCONT	Redémarrage d'un processus temporairement arrêté.
19	SIGSTOP	Arrêt temporaire d'un processus (ne peut être ignoré).
20	SIGTSTP	Arrêt temporaire d'un processus (Ctrl + Z). Peut être ignoré.
21	SIGTTIN	Signal envoyé à un processus lorsqu'il tente de lire depuis son terminal alors qu'il est en tâche de fond (un processus en tâche de fond ne peut pas lire/écrire via un terminal).
22	SIGTTOU	Signal envoyé à un processus lorsqu'il tente d'écrire sur son terminal alors qu'il est en tâche de fond (un processus en tâche de fond ne peut pas lire/écrire via un terminal).
23	SIGURG	Indique à un processus que des données dites « hors-bande » sont arrivées sur un socket TCP/IP. Les données « hors-bande » sont à un flux réseau ce que les caractères de contrôle sont à une chaîne de caractères.
24	SIGXCPU	Émis par le kernel quand un processus dépasse le temps d'utilisation CPU qui lui a été alloué.

By **TME520** (TME520)
cheatography.com/tme520/
sysadmin.tme520.net

Published 9th April, 2015.
Last updated 9th April, 2015.

Les signaux classiques (cont)		
25	SIGXFSZ	Émis par le kernel quand un processus tente de créer un fichier trop grand.
26	SIGVTALRM	Indique le déclenchement d'un timer virtuel. Comparable à une sonnerie de réveil pour processus.
27	SIGPROF	Indique le déclenchement d'un timer dit « de profiling ». On fait du *profiling* sur un processus quand on veut en améliorer les performances.
28	SIGWINCH	La fenêtre du terminal a été redimensionnée.
29	SIGIO	Ne pas confondre avec SIGIOT, l'équivalent sous UNIX System V du SIGABRT de Linux. Indique à un processus qu'il peut effectuer une opération d'entrée/sortie (I/O).
30	SIGPWR	Indique un problème d'alimentation. Utile notamment sur systèmes mobiles ou alimentés via un onduleur suite à une coupure du secteur.
31	SIGSYS	Le programme a tenté d'appeler une fonction système invalide.

5 exemples		
Redémarrer un daemon	/bin/kill -SIGHUP PID	/bin/kill -1 PID
Arrêter proprement un processus	/bin/kill -SIGTERM PID	/bin/kill -15 PID
Arrêter brutalement un processus	/bin/kill -SIGKILL PID	/bin/kill -9 PID
Mettre un processus en pause	/bin/kill -SIGSTOP PID	/bin/kill -19 PID
Sortir un processus de pause	/bin/kill -SIGCONT PID	/bin/kill -18 PID

By **TME520** (TME520)
cheatography.com/tme520/
sysadmin.tme520.net

Published 9th April, 2015.
Last updated 9th April, 2015.