Sygnały

Procesy komunikują się z jadrem systemu, a także między sobą, aby koordynować swoją działalność. Linux wspiera kilka mechanizmów komunikacji zwanych IPC (*Inter-Process Communication mechanisms*). Jednym z nich są **sygnały**, zwane inaczej przerwaniami programowymi.

Sygnały mogą być generowane bezpośrednio przez użytkownika (funkcja kill()), może wysyłać je jądro oraz procesy między sobą (funkcja systemowa kill()). Dodatkowo pewne znaki z terminala powodują wygenerowanie sygnałów. Na przykład, na każdym terminalu istnieje tak zwany znak przerwania (ang. *interrupt character*) i znak zakończenia (ang. *quit character*). Znak przerwania (Ctrl-C) służy do zakończenia bieżącego procesu (wygenerowanie sygnału SIGINT). Wygenerowanie znaku zakończenia (Ctrl-\) powoduje wysłanie sygnału SIGQUIT powodującego zakończenie wykonywania bieżącego procesu z zapisaniem obrazu pamięci.

Istnieją oczywiście pewne ograniczenia - proces może wysyłać je tylko do procesów mających tego samego właściciela oraz z tej samej grupy (te same uid i gid). Bez ograniczeń może to czynić jedynie jądro i administrator. Jedynym procesem, który nie odbiera sygnałów jest **init** (posiada pid równy 1).

Sygnały są mechanizmem asynchronicznym - proces nie wie z góry, kiedy sygnał może nadejść i głównym ich zadaniem jest informowanie procesu o zaistnieniu w systemie wyjątkowej sytuacji (np.: spadek napięcia w sieci). Ponadto są wykorzystywane przez powłokę do kontroli pracy swoich procesów potomnych.

Implementacja struktur danych związanych z sygnałami, ustalenie ich liczby, numerów i nazw znajduje się w pliku nagłówkowym signal.h. Natomiast informacje o tym, jakie sygnały dany proces otrzymał, które z nich chce ignorować, a które obsłużyć i w jaki sposób zapamiętane są w specjalnych polach struktury task_struct. (szczegółów szukaj w opisie struktur danych procesu).

Ilość dostępnych sygnałów jest ograniczona przez rozmiar słowa procesora. Stąd komputery 32-bitowe mają maksymalną ilość 32 sygnałów, podczas gdy maszyny 64-bitowe mogą mieć zaimplementowane do 64 sygnałów.

Poniżej lista sygnałów zamykających procesu w systemie Linux:

SIGTERM (15). The SIGTERM signal is a generic signal used to cause program termination. Unlike SIGKILL, this signal can be blocked, handled, and ignored. It is the normal way to politely ask a program to terminate. The shell command kill generates SIGTERM by default.

SIGINT (2). The SIGINT (“program interrupt”) signal is sent when the user types the INTR character (Ctrl+C).

SIGQUIT (3). The SIGQUIT signal is similar to SIGINT, except that it’s controlled by a different key—the QUIT character, usually Ctrl+\ and produces a core dump when it terminates the process, just like a program error signal. You can think of this as a program error condition “detected” by the user. Certain kinds of cleanups are best omitted in handling SIGQUIT. For example, if the program creates temporary files, it should handle the other termination requests by deleting the temporary files. But it is better for SIGQUIT not to delete them, so that the user can examine them in conjunction with the core dump.

SIGKILL (9). The SIGKILL signal is used to cause immediate program termination. It cannot be handled or ignored, and is therefore always fatal. It is also not possible to block this signal. This signal is usually generated only by explicit request. Since it cannot be handled, you should generate it only as a last resort, after first trying a less drastic method such as Ctrl+C or SIGTERM. If a process does not respond to any other termination signals, sending it a SIGKILL signal will almost always cause it to go away. In fact, if SIGKILL fails to terminate a process, that by itself constitutes an operating system bug which you should report. The system will generate SIGKILL for a process itself under some unusual conditions where the program cannot possibly continue to run (even to run a signal handler).

SIGHUP (1). The SIGHUP ("hang-up") signal is used to report that the user's terminal is disconnected, perhaps because a network or telephone connection was broken. For more information about this, see Control Modes. This signal is also used to report the termination of the controlling process on a terminal to jobs associated with that session; this termination effectively disconnects all processes in the session from the controlling terminal.