

```
user@centos8:~  
File Edit View Search Terminal Help  
[user@centos8 ~]$ gedit  
Terminated  
[user@centos8 ~]$  
user@centos8:~  
File Edit View Search Terminal Help  
[user@centos8 ~]$ ps -ef | grep gedit  
user      3306   3219    1 10:25 pts/0    00:00:00 gedit  
user      3362   3312    0 10:26 pts/1    00:00:00 grep --colo  
dit  
[user@centos8 ~]$ kill 3306
```

kill позволяет нам посылать сигналы. Судя по названию – в основном, чтобы убивать процессы. Для примера, откроем эмулятор терминала и выполним команду gedit - откроется блокнот. Потом найдём идентификатор процесса блокнота с помощью ps -ef | grep gedit и используем команду kill с нужным pid. И вот - блокнот закрылся.

```
user@centos8:~  
File Edit View Search Terminal Help  
[user@centos8 ~]$ 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
```

Давайте запустим kill -l и посмотрим список сигналов. До этого мы написали просто kill и номер процесса, из-за чего процессу послался сигнал по умолчанию – 15) SIGTERM. Это мягкий сигнал, который даёт процессу время закончить свои дела, прежде чем умереть. За это время процесс успевает попрощаться с дочерними и родительскими процессами. После смерти, дочерние процессы становятся процессами-сиротами, а их родителем становится процесс с номером 1.

```
user@centos8:~$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
user         7297    6985  0 11:14 pts/1        00:00:00 bash
user         8129    7297  0 11:23 pts/1        00:00:00 ps -f
[user@centos8 ~]$ gedit &
[1] 8149

user@centos8:~$ ps -ef | grep gedit
user         8149    7297  1 11:23 pts/1        00:00:00 gedit
user         8222    8167  0 11:24 pts/0        00:00:00 grep --color=aut
```

Для примера, откроем окно эмулятора терминала, там у нас появится новая bash сессия – ps -f. В этом окне я запущу блокнот с амперсандом в конце - gedit &. Амперсанд нужен, чтобы выполнить команду в фоне, а не быть зависимым от эмулятора терминала. При запуске мы увидели id процесса блокнота. Можно посмотреть – ps -ef | grep gedit, где видно, что у процесса gedit родительским процессом является bash сессия в этом эмуляторе терминала.

```
user@centos8:~$ ps -ef | grep gedit
user         8149    7297  1 11:23 pts/1        00:00:00 gedit
user         8222    8167  0 11:24 pts/0        00:00:00 grep --color=auto gedit
[user@centos8 ~]$ kill 7297
[user@centos8 ~]$ ps -ef | grep gedit
user         8149    2508  0 11:23 ?              00:00:00 gedit
user         8731    8167  0 11:29 pts/0        00:00:00 grep --color=auto gedit
[user@centos8 ~]$ ps 2508
  PID TTY          STAT       TIME COMMAND
 2508 ?        Ss           0:00 /usr/lib/systemd/systemd --user
[user@centos8 ~]$ ps 1
  PID TTY          STAT       TIME COMMAND
    1 ?        Ss           0:01 /usr/lib/systemd/systemd --switched-root --system
```

Теперь давайте избавимся от родительского процесса – kill ppid. Блокнот все ещё запущен. Найдём его – ps -ef | grep gedit – и у блокнота изменился родительский процесс. Но я говорил, что процессы сироты забирает себе процесс с номером 1, а тут другой ppid. Если посмотреть этот процесс – ps ppid - и первой процесс – ps 1, то команды будут похожи – systemd. В большинстве unix-подобных систем сирот на себя берёт первый процесс, но то что мы видим сейчас – связано с изменениями, которые происходят в последние лет 10. Во первых, это пока не затронуло все дистрибутивы, во вторых, это связано именно с пользовательскими сессиями и не так актуально на серверах. Про systemd мы ещё поговорим, я же просто объяснил, что бывает с дочерними процессами.

```
user@centos8:~  
File Edit View Search Terminal Help  
[user@centos8 ~]$ ps -ef | grep gedit  
user      8149   2508   0 11:23 ?          00:00:00 gedit  
user      9098   8167   0 11:34 pts/0    00:00:00 grep --  
[user@centos8 ~]$ kill -9 8149  
[user@centos8 ~]$ kill -SIGKILL 8149
```

Вернёмся к сигналам. Когда мы посылаем SIGTERM, процесс сам отвечает за своё завершение. Но что если процесс завис, ни на что не реагирует? На самом деле, в таких случаях лучше разобраться, с чем это связано. Есть способ по жёсткому избавиться от процесса, используя сигнал – 9) SIGKILL. Но это очень опасный сигнал – резкое убийство процесса, без возможности завершить все дела, может повредить базу данных, файловую систему и т.п. Поэтому нужно быть крайне аккуратным с этим сигналом и лучше постараться найти причину проблемы и попытаться решить её, а не убивать процесс. Но всё же, если нет другого выхода – посылаем kill -9 pid. Можно ещё вместо номеров использовать сами сигналы – kill -SIGKILL pid.

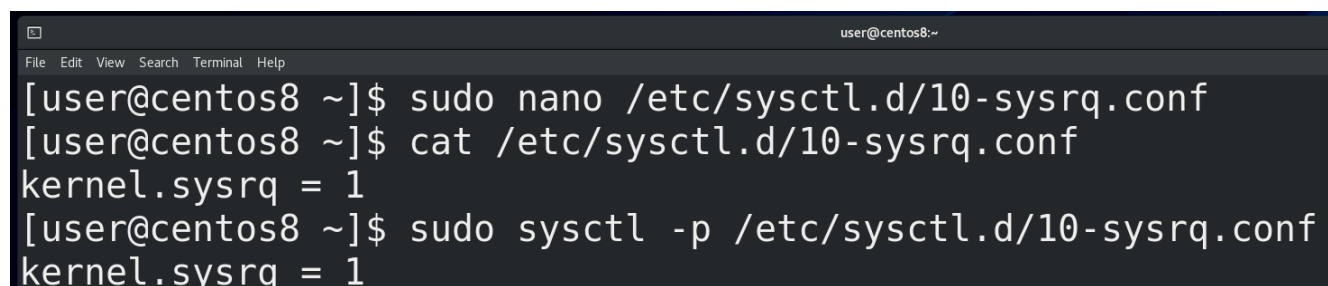
```
user@centos8:~  
File Edit View Search Terminal Help  
[user@centos8 ~]$ gedit &  
[1] 9411  
[user@centos8 ~]$ pkill -19 gedit  
  
[1]+  Stopped                               gedit  
[user@centos8 ~]$ pkill -18 gedit
```

Большинство сигналов нужны не столько администраторам, сколько разработчикам. Но пара занятных сигналов всё же есть, допустим - 19) SIGSTOP - о котором я говорил в прошлый раз. Потренируемся на том же блокноте. Чтобы постоянно не искать id процесса, я могу использовать команду pkill. Она сама ищет процесс по шаблону, но нужно быть осторожным, потому что в некоторых случаях у разных процессов могут быть совпадения и это может привести к плохим последствиям. Но так как блокнот запущен один, я пишу - pkill -19 gedit. И теперь блокнот ни на что не реагирует, как будто завис. Если я хочу, чтобы он продолжил работу, я посылаю сигнал 18) SIGCONT – от слова continue – и он опять продолжает работать.

И так, когда у вас начинает зависать система, смотрим top или htop, сортируем по cpi или метогу, а потом решаем, что делать с процессом – пытаемся найти причину и решить. Допустим, у вас шёл бэкап базы данных на внешний сервер, а он отвалился от сети и теперь база данных начала грузить процессы. Легче всего будет вернуть в работу бэкап сервер. Это просто пример, возможно не самый хороший, просто объясняющий, что делать. Если процесс не такой

важный и это какая-то мелочь, типа блокнота, можно попытаться ему послать сигнал с помощью kill, для начала тот же SIGTERM. Старайтесь избегать использования SIGKILL, это крайняя мера.

Если у вас зависает графический интерфейс, попробуйте открыть виртуальный терминал – ctrl+alt+f3, f4 и т.п. и решайте проблему оттуда. Виртуальный терминал особо ресурсы не расходует, и, в случае тормозов, он работает лучше графического интерфейса. Ещё одна проблема, которая может быть – ваша система намертво зависает и не реагирует абсолютно ни на что. Зачастую, это связано с проблемой нехватки памяти – возможно какой-то багованный софт съел всю оперативную память, отчего вся система зависла. Это называется OOM – out of memory – и различные компании предлагают свои превентивные меры против этой проблемы, достаточно погуглить oom killer linux. Но это всё превентивные меры, а если вы всё же столкнулись с проблемой, не спешите перезагружать систему. Вам поможет магическая кнопка [sysrq](#). На современных клавиатурах её не всегда пишут, но обычно это та же клавиша PrtSc. Эта клавиша позволяет напрямую посылать какие-то команды ядру.

A screenshot of a terminal window with a dark background. The window title is "user@centos8:~". The menu bar shows "File Edit View Search Terminal Help". The terminal shows the following commands and output:

```
[user@centos8 ~]$ sudo nano /etc/sysctl.d/10-sysrq.conf
[user@centos8 ~]$ cat /etc/sysctl.d/10-sysrq.conf
kernel.sysrq = 1
[user@centos8 ~]$ sudo sysctl -p /etc/sysctl.d/10-sysrq.conf
kernel.sysrq = 1
```

Например, чтобы решить проблему out of memory, стоит нажать Alt+PrtSc+f, тогда ядро избавится от проблемного процесса, чтобы освободить память. Но это нужно предварительно настроить. Для этого создадим файл внутри директории /etc/sysctl.d с расширением .conf - sudo nano /etc/sysctl.d/10-sysrq.conf. Напишем в этом файле - kernel.sysrq = 1. Сохраним, затем выполним команду - sudo sysctl -p /etc/sysctl.d/10-sysrq.conf. Есть и другие способы, допустим, установка демона oom-killer (earlyoom, systemd-oom и т.п.), который будет решать проблему без ручного вмешательства. Но это отдельная тема.

Таким образом, мы разобрались с тем, где найти информацию по процессам, немного углубились в теорию и разобрали, как управлять процессами с помощью сигналов, какие есть превентивные меры – ulimit, oom killer-ы - и что делать, когда система тормозит или вообще зависла.