Команды kill и killall в Linux

Инструкция о том, как команд правильно использовать команды kill и killall.

Эта инструкция — часть курса «Linux: горячие клавиши и команды».

Смотреть весь курс

Введение

Мы привыкли, что Linux — семейство Unix-подобных операционных систем на базе одного ядра — отличается устойчивостью и стабильностью. Но даже в таких ОС в повседневной жизни можно столкнуться с ситуацией, когда программы начинают зависать, нестабильно отвечать или просто перестают работать. Например, они настолько медленно реагируют, что их нельзя закрыть обычным способом. Также не отвечающие приложения нельзя перезапустить, потому что исходный процесс приложения не завершается окончательно.

Первым решением, к которому обычно прибегают,— перезагрузка системы. Но еще в таких случаях можно «убить» запущенную программу при помощи команд kill и killall. Дальше мы подробно рассмотрим, как их правильно использовать.

Процессом мы называем любую выполняющуюся программу. Иначе еще его можно назвать функционирующим экземпляром программы. Если вы запустили два окна с одним и тем же приложением — это будет два процесса, а не один.

PID процесса

В Linux у процесса есть уникальный идентификатор Process ID или сокращенно PID. При создании каждого процесса ему автоматически присваивается PID.

Отношения между процессами в системе

Родителем называется процесс, который отвечает за создание дочернего процесса. Если родительский процесс завершится раньше, чем дочерний, то тогда «потомок» привяжется к init, который был создан первый при загрузке (но также есть опция переназначить дочерний процесс другому).

Если же дочерний процесс прекратит свою работу раньше, он останется без родителя до тех пор, пока «предок» не получит информацию про него или не сообщит ядру, что такая информация не требуется. Тогда ресурсы, используемые дочерним процессом, освобождаются.

Возможные значения PID

В зависимости от значения PID, который мы в дальнейшем будем передавать kill, сигнал будет отправляться разной группе процессов. Разберем, как именно значение PID влияет на отправку сигнала:

- **PID > 0.** Сигнал будет послан процессу, чей идентификатор соответствует введенному значению.
- **PID = 0**. Все процессы, входящие в текущую группу, получают этот сигнал.
- **PID = −1**. В этом случае сигнал будет послан всем пользовательским процессам (если команду вызывает root, то сигнал идет всем процессам кроме init и самого kill). Для этого существует специальный идентификатор пользователя (UID).
- **PID < −1**. Сигнал отправляется процессам, включенным в группу с GID (идентификатор группы), соответствующим абсолютному значению PID.

Как убить зависший процесс

Чтобы завершить конкретный процесс, мы должны понять, какой у него номер. Для этого обратимся к вспомогательным командам. Если мы хотим посмотреть работающие процессы, можно запустить удобный интерфейс командой top. Сначала она отобразит всю статистику по системе (нагрузку и общее количество задач), а после покажет все процессы, отсортированные в реальном времени по их нагрузке на процессор. Но такой способ неудобен для поиска конкретного PID.

- **ps** – команда, выводящая перечень действующих процессов с дополнительными сведениями о них.
- **grep** выполняется вместе с ps и ищет по списку, полученному от ps.

Чтобы извлечь все процессы, нужно написать в терминале:

```
ps axu
```

Зачастую этот список достаточно длинный и найти PID заданного процесса затруднительно. Тогда стоит обратиться к команде grep. Для поиска процесса, имеющего конкретное имя, нужно выполнить:

```
ps axu | grep bash
```

Команда осуществит поиск по результату, выданному командой ps, и на экране мы увидим только строки, содержащие bash.

Если такого процесса не найдется, то в качестве ответа мы получим сам grep: это происходит из-за того, что параметром указано слово bash и grep находит сам себя.

Если же такой процесс существует, то мы увидим подобный вывод:

```
root 447 0.0 0.4 7932 4708 tty1 S 11:26 0:00 -bash
root 477 0.0 0.0 6268 640 tty1 S+ 11:27 0:00 grep bash
```

Нам здесь важна первая строчка. Число после имени пользователя и будет являться уникальным идентификатором.

Для нахождения PID еще можно попробовать команду pgrep – утилиту, которая просматривает активные процессы и выдает PID тех, чьи атрибуты удовлетворяют запросу:

```
pgrep firefox
```

Еще один вспомогательный способ определения PID: pidof – команда, принимающая на вход название:

```
pidof gcalctool
```

Если исполняемая программа задействует не один процесс, то команда в результате выведет все идентификаторы.

Команда kill

Теперь, зная PID, процесс можно принудительно остановить. Применим команду kill, передав вместе с ней номер:

kill шлет сигнал: не указывая аргументом, какой именно сигнал мы хотим отправить, по умолчанию идет SIGTERM. Сигнал сообщает потребность в завершении процесса. Кроме того, процессу предоставляется возможность потенциально остановиться без ошибок, освободив ресурсы. Сложность заключается в том, что сигнал может быть проигнорирован. У всех сигналов есть номер, у SIGTERM, например, 15. Чтобы узнать список всех возможных сигналов и их номеров, нужно ввести следующее:

```
kill -l
```

Но все-таки SIGTERM не гарантирует остановку процесса в случае блокировки сигнала или его перехвате. Чтобы наверняка прервать работу процесса без угрозы игнорирования, нужно послать SIGKILL (номер 9) – он немедленно прерывает процесс:

```
kill -9 279
```

После выполнения несохраненный прогресс будет утерян. Тут обратим внимание, что команда kill может быть внутренней (часть встроенной функции оболочки) или внешней (находящейся по адресу /bin/kill). Независимо от этого синтаксис остается одинаковым. Чтобы найти все упоминания kill в системе, выполните следующее:

```
type -a kill
```

Примечание: с помощью kill можно убивать сразу несколько процессов. Для этого указываем их PID подряд через пробел:

```
kill -9 267 315 442
```

Либо если вы не хотите указывать каждый PID по отдельности вручную, можно сделать так:

```
kill -9 $(pidof gcalctool)
```

Утилита pkill

pkill – оболочка для kill. Она позволяет применять регулярные выражений или другие критерии для нахождения процессов. У нее такой же синтаксис, как и у команды kill, но в качестве PID можно передать имя процесса (или часть его имени). Утилита работает следующим образом: она просматривает директорию proc и находит идентификатор первого процесса с подходящим именем. После этого отправляет этому процессу SIGTERM.

```
pkill ping
```

Аналогично мы можем задать тип сигнала:

Сейчас вы на 7 статье курса
01 · 10 минут
Командная строка Linux: краткий курс для начинающих
Полный список курса (9)

Еще раз нужно отметить, что эта утилита убивает только первый найденный процесс. Поэтому если у вас открыто несколько вкладок в браузере, то pkill закроет только одну из них.
Также у команды pkill есть и другая опция: если ввести флаг -u, то после мы можем написать имя пользователя (или его ID).

```
pkill -u user
```

Так мы остановим первый процесс, заданный этим пользователем.

Команда killall

killall — команда, которая прерывает процессы с одним и тем же названием. Если у вас будет запущено несколько окон с одинаковыми приложениями (а мы уже знаем, что это будут разные процессы), killall уничтожит все. Еще одно отличие заключается в том, что здесь нет необходимости знать PID (команда ищет все подходящие в папке /proc). Если мы аналогично одному из примеров, хотим закрыть окна калькулятора, нужно написать:

```
killall gcalctool
```

Без аргументов killall также отправляет сигнал SIGTERM. Для отправки другого сигнала нужно явно это прописать:

```
killall -s 1 gcalctool
```

Не всегда удается помнить правила использований всех команд. Чтобы получить справку по их применению, можно применить команду man — она выдаст информацию про любую команду:

```
man killall
```

Как проверить, что процесс убит

Чтобы убедиться, что процесс был действительно отменен, надо воспользоваться одним из способов:

```
ps aux | grep
```

```
pidof
```

```
pgrep
```

При успешном окончании эти команды ничего не выведут.

Подробности о завершении

Права пользователя root

Обычному пользователю доступно завершить только собственные процессы. Это объясняется тем, что для передачи сигналов может не хватать прав. Однако root может посылать сигналы и чужим процессам.

Чтобы прервать любой процесс (неважно, был ли он запущен пользователем root или любым другим пользователем), kill и killall должны быть выполнены от суперпользователя. Например, в Ubuntu это делается добавлением слова sudo:

```
sudo kill 279
```

Также только у root есть права на отмену процесса системного уровня.

Использование виртуальной консоли

Допустим, вам необходимо остановить зависнувший процесс, но у вас не получается открыть терминал. Тогда стоит зайти в виртуальную консоль, нажав клавиши Ctrl + Alt + F1, авторизоваться (ввести логин и пароль) и исполнить скрипт из нее. Чтобы перейти обратно, нужно применить Ctrl + Alt + F7.

Отправка сигналов в Linux

Теперь подробнее разберем, что такое сигналы и как они работают. В Linux взаимодействие с процессами осуществляется через программные прерывания – сигналы. Система или сам пользователь передают их, вводя команды. После того как процесс получил сигнал об окончании, ему желательно осуществить некоторые подготовительные шаги.

Чтобы полностью завершить выполнение процесса, необходимо также прервать дочерние элементы, удалить временные файлы и другие дополнительные действия. Однако процесс может отреагировать не на каждый сигнал. Сигнал, остановивший процесс, выполнил свою главную задачу. Разберем сигналы, еще не описанные до этого.

- SIGINT (2) – самый безопасный сигнал для отмены процесса. После этого процесс правильно завершается, и управление возвращается. Также его можно отправить с помощью клавиш Ctrl + C.
- SIGQUIT (3) – этот сигнал также сообщает программе об окончании. Программа либо выполняет корректное завершение, либо игнорирует сигнал. В отличие от SIGINT, этот сигнал генерирует содержимое рабочей памяти одного процесса. Можно послать с помощью клавиатуры, используя сочетание Ctrl + /.
- SIGHUP (1) перезагружает процесс, а также используется для перезагрузки файлов конфигурации и открытия или закрытия файлов журнала. Чаще посылается системой при потере соединения с интернетом и сообщает процессу об отсутствии связи с управляющим терминалом.

Чтобы процесс завершился верно, порты и сокеты освобождаются и закрываются, а также удаляются временные файлы. Поэтому не стоит использовать сразу радикальные меры – передавать SIGKILL. Сначала пытаемся завершить процесс в следующем рекомендуемом порядке:

1. SIGINT (Ctrl + C).
2. SIGTERM – аккуратно останавливает процесс.
3. SIGKILL – последнее средство для завершения.

До этого мы уже привели несколько примеров, как можно задать тип сигнала. Подведем итоги:

1. Номер (-1 или -s 1).
2. Полное название (-SIGHUP или -s SIGHUP).
3. Сокращенное название, без префикса «SIG» (-HUP или -s HUP).

Настройка процессов

Перезагрузка процессов с помощью kill

Мы уже знаем, что команда kill и сигнал SIGHUP объявляют, что требуется обновить настройки. Чтобы подробнее разобрать, как это происходит, рассмотрим Nginx. Чтобы перезапустить сервер, нужно послать сигнал главному процессу. Идентификатор ведущего процесса находится в файле nginx.pid, который обычно расположен в каталоге /var/run.

Теперь воспользуемся командой **cat**, которая последовательно считывает содержимое файла и записывает его в стандартный поток вывода:

```
cat /var/run/nginx.pid
```

После этого мы увидим искомый PID. Теперь осталось обновить настройки, написав:

```
sudo kill -1 740
```

Эту команду можно запустить от root или с привилегиями sudo.

Управление приоритетами процессов

Иногда требуется изменить приоритет у процесса в серверной среде, ведь определенные процессы могут быть очень важны, а другие исполняться на излишках предоставляемых ресурсов.

Для контроля приоритетов у Linux есть понятие вежливости. Самые приоритетные задачи считаются менее вежливыми, потому что они не делятся ресурсами. Тогда процессы с низким приоритетом считаются более вежливыми, потому им требуется минимум ресурсов.

В выводе команды top есть столбец NI, целью которого является отобразить значение вежливости. Вежливость может варьироваться от −20 (наибольший приоритет) до 20 (соответственно, наименьший).

Если по каким-то причинам мы хотим посмотреть на программу с определенным значением вежливости, то мы воспользуемся nice:

```
nice -n 10
```

А для того чтобы изменить значение вежливости у уже выполняемой программы, воспользуемся инструментом renice, которому передаем PID:

```
renice 11
```

Заключение

Команды kill и killall — мощные инструменты в Linux. Они позволяют прекратить выполнение запущенных процессов. Они представляют собой неотъемлемую частью администрирования Linux и применяются в различных сценариях, таких как прекращение работы процессов, вызов перезапуска служб и многих других.

Несмотря на их эффективность, необходимо быть осторожным при написании команд kill и killall, так как неправильное обращение может привести к удалению данных и неожиданным последствиям. Рекомендуем ознакомиться с руководством по применению этих команд, чтобы убедиться, что они используются правильно и безопасно.

9 минут
Перенаправление ввода/вывода в Linux
Предыдущий материал

12 минут
Команда ifconfig в Linux: установка, синтаксис и примеры···
Следующий материал

Читайте также:

Инструкция
Как создать 100 серверов в облаке за минуту? Работа с OpenStack клиентом
24 января 2024

Инструкция
Как создать веб-приложение на базе Telegram Mini Apps
1 декабря 2023

Инструкция
Что делает команда chmod и как ее использовать в Linux
29 ноября 2023