Управление процессами

Процесс это – совокупность программного кода и данных, загруженных в память ЭВМ.

На первый взгляд процесс – это запущенная программа (приложение) или команда. Но это не совсем так. Некоторые приложения могут создавать несколько процессов одновременно.

Код процесса не обязательно должен выполняться в текущий момент времени, так как процесс может находиться в состоянии спящего. В этом случае выполнение кода такого процесса приостановлено. Существует всего 3 состояния, в которых может находиться процесс:

Работающий процесс – в данный момент код процесса выполняется.

Спящий процесс – в данный момент код процесса не выполняется в ожидании какого либо события (нажатия клавиши на клавиатуре, поступление данных из сети и т.д.)

Процесс-зомби – сам процесс уже не существует, его код и данные выгружены из оперативной памяти, но запись в таблице процессов остается по тем или иным причинам.

Каждому процессу в системе назначаются числовые идентификаторы (личные номера) в диапазоне от 1 до 65535 (**PID – Process Identifier – идентификатор процесса**) и идентификаторы родительского процесса (**PPID – Parent Process Identifier – идентификатор родительского процесса**). PID является именем процесса, по которому мы можем адресовать процесс в операционной системе при использовании различных средств просмотра и управления процессами. PPID определяет родственные отношения между процессами, которые в значительной степени определяют его свойства и возможности.

Для просмотра списка процессов в Linux существует команда ***ps***. Формат команды следующий:

$ ps *[PID] [options]*

```
/home/larry# ps
PID TT STAT  TIME COMMAND
24  3  S     0:03 (bash)
161 3  R     0:00 ps
/home/larry#
```

По умолчанию команда ps выводит список только тех процессов, которые принадлежат запустившему её пользователю. Чтобы посмотреть все исполняющиеся в системе процессы, нужно подать команду ps -a. **Номера процессов** (process ID, или PID), указанные в первой колонке, являются уникальными номерами, которые система присваивает каждому работающему процессу.

Последняя колонка, озаглавленная COMMAND, указывает имя работающей команды.

Данная команда имеет много параметров, о которых вы можете прочитать в руководстве (man ps).

Параметр	Описание
-a	отобразить все процессы, связанных с терминалом (отображаются процессы всех пользователей)
-e	отобразить все процессы
-t список терминалов	отобразить процессы, связанные с терминалами
-u идентификаторы пользователей	отобразить процессы, связанные с данными идентификаторыми
-g идентификаторы групп	отобразить процессы, связанные с данными идентификаторыми групп
-x	отобразить все процессы, не связанные с терминалом

PID, PPID – идентификатор процесса и его родителя.

%CPU – доля процессорного времени, выделенная процессу.

%MEM – процент используемой оперативной памяти.

VSZ – виртуальный размер процесса.

TTY – управляющий терминал.

STAT– статус процесса:

- R – выполняется;
- S – спит;
- Z – зомби;
- < – Повышенный приоритет;
- + – Находится в интерактивном режиме.

START – время запуска.

TIME – время исполнения на процессоре.

Команда ps делает моментальный снимок процессов в текущий момент. Вы также можете просмотреть и другую информацию о каждом процессе. Опция --forest позволяет легко просмотреть иерархию процессов и даст вам представление о том, как различные процессы в системе взаимосвязаны между собой. Если один процесс запускает другой процесс, то запущенный будет называться его потомком. В выводе --forest, родители находятся слева, а потомки появляются как ветки справа:

```
$ ps x --forest
PID       TTY      STAT   TIME COMMAND
927       pts/1    S      0:00 bash
6690      pts/1    S      0:00  \_ bash
26909     pts/1    R      0:00      \_ ps x --forest
19930     pts/4    S      0:01 bash
```

Команда top - динамически выводит состояние процессов и их активность в реальном режиме времени.

```
$ top
10:02pm  up 19 days,  6:24,  8 users,  load average: 0.04, 0.05, 0.00
75 processes: 74 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:           1.3% user,           2.5% system,   0.0% nice,     96.0% idle
Mem:    256020K av,    226580K used,  29440K free,   0K shrd,       3804K buff
Swap:   136544K av,    80256K used,   56288K free            101760K cached

PID     USER          PRI     NI      SIZE    RSS     SHARE   STAT    LIB     %CPU    %MEM    TIME    COMMAND
628     root          16      0       213M    31M     2304    S       0       1.9     12.5    91:43   X
26934   chouser       17      0       1272    1272    1076    R       0       1.1     0.4     0:00    top
652     chouser       11      0       12016   8840    1604    S       0       0.5     3.4     3:52    gn-term
641     chouser       9       0       2936    2808    1416    S       0       0.1     1.0     2:13    sawfish
```

Родителем всех процессов в системе является процесс *init*. Его PID всегда 1, PPID – 0. Всю таблицу процессов можно представить себе в виде дерева, в котором корнем будет процесс *init*. Этот процесс хоть и не является частью ядра, но выполняет в системе очень важную роль – определяет текущий уровень инициализации системы и следит, чтобы были запущены программы, позволяющие пользователю общаться с компьютером.

Процессы, имена которых заключены в квадратные скобки, например *“[keventd]”* – это процессы ядра. Эти процессы управляют работой системы, а точнее такими ее частями, как менеджер памяти, планировщик времени процессора, менеджеры внешних устройств и так далее.

Во время работы процесса, ядро контролирует его состояние, и в случае возникновения непредвиденной ситуации управляет процессом с помощью посылки ему сигнала. **Сигнал** – это простейший способ межпроцессорного (то есть между процессами) взаимодействия. Существует несколько типов сигналов. Для каждого из типов предусмотрено действие по умолчанию. Процесс может воспользоваться действием по умолчанию, или, если у него есть обработчик сигнала, то он может перехватить и обработать или игнорировать сигнал. Сигналы **SIGKILL (**Сигнал, при получении которого выполнение процесса прекращается**) и SIGSTOP (**Сигнал отправляется всем процессам текущей группы при нажатии пользователем клавиш <CTRL>+<Z>. Получение сигнала вызывает останов выполнения процесса**)** невозможно ни перехватить, ни игнорировать.

Два сигнала—номер 9 (KILL) и 19 (STOP)—всегда обрабатывает система. Первый из них нужен для того, чтобы убить процесс наверняка. Сигнал STOP *приостанавливает* процесс: в таком состоянии процесс не удаляется из таблицы процессов, но и не выполняется до тех пор, пока не получит сигнал 18 (CONT) — после чего продолжит работу. В командной оболочке Linux сигнал STOP можно передать активному процессу с помощью управляющей последовательности *Ctrl–Z*.

Сигнал номер 15 (TERM) служит для прерывания работы задания.

При **прерывании** (interrupt) задания процесс погибает. Прерывание заданий обычно осуществляется управляющей последовательностью *Ctrl–C*. Восстановить прерванное задание никаким образом невозможно.

$kill -SIGNAL pid – посылает сигнал процессу с идентификатором pid. Если сигнал не указан, команда посылает процессу сигнал SIGTERM.

При обычном запуске процесс работает на переднем плане, то есть процесс "привязывается" к терминалу, с которого он запущен, воспринимая ввод с этого терминала и осуществляя на него вывод. Но можно запустить процесс в фоновом режиме, когда он не связан с терминалом, для чего в конце командной строки запуска программы добавляют символ &.

В оболочке bash имеются две встроенные команды, которые служат для перевода процессов на передний план или возврата их в фоновый режим. Команда **fg** переводит указанный в аргументе процесс на передний план, а команда **bg** — переводит процесс в фоновый режим.

Каждому процессу при запуске устанавливается определенный приоритет, который имеет значение от -20 до +20, где +20 - самый низкий. Приоритет нового процесса равен приоритету процесса-родителя. Для изменения приоритета запускаемой программы существует утилита nice. Пример ее использования:

nice [- adnice] command [args]

где adnice — значение (от –20 до +19), добавляемое к значению nice процесса-родителя. Отрицательные значения может устанавливать только суперпользователь. Если опция adnice не задана, то по умолчанию для процесса-потомка устанавливается значение nice, увеличенное на 10 по сравнению со значением nice родительского процесса.

Команда renice служит для изменения значения nice для уже выполняющихся процессов. Суперпользователь может изменить приоритет любого процесса в системе. Другие пользователи могут изменять значение приоритета только для тех процессов, для которых данный пользователь является владельцем. При этом обычный пользователь может только уменьшить значение приоритета.