4. Управление процессами

Первым делом научимся определять, какие процессы в системе запущены. Для этого в Linux (как и во всех UNIX-системах) имеется команда `ps`. Если ее запустить без всяких параметров, то она выдает список процессов, запущенных в текущей сессии. Если вы хотите увидеть список всех процессов, запущенных в системе, надо задать ту же команду с параметром `-ax`.

4.1. Команда ps

Когда я заглянул в man-страницу, посвященную команде `ps`, я был поражен, как много у нее разных опций. Как оказалось, GNU-версия этой программы, входящая в состав Linux, поддерживает опции в стиле трех разных типов UNIX. Опции в стиле Unix98 состоят из одного или нескольких символов, перед которым(и) должен стоять дефис. Опции в стиле BSD имеют аналогичный вид, только используются без дефиса. Опции, характерные только для GNU-версии представляют собой слово, перед которым должно стоять два дефиса. Их нельзя объединять, как однобуквенные опции двух предшествующих типов. Таким образом, существует три равноправных формата задания этой команды:

```
ps [-опции]
ps [опции]
ps [-- длинное_имя_опции [-- длинное_имя_опции] ...]
```

При этом опции разных типов нельзя употреблять в одной команде. Дадим краткую характеристику наиболее важных опций.

Первая группа опций регулирует вывод команды. Независимо от наличия опций этой группы команда `ps` выдает для каждого процесса отдельную строку, но содержимое этой строки может быть разным. В зависимости от заданных опций могут присутствовать следующие поля:

- USER — имя владельца процесса;
- PID — идентификатор процесса в системе;
- PPID — идентификатор родительского процесса;
- %CPU — доля времени центрального процессора (в процентах), выделенного данному процессу;
- %MEM — доля реальной памяти (в процентах), используемая данным процессом;
- VSZ — виртуальный размер процесса (в килобайтах);
- RSS — размер резидентного набора (количество 1K-страниц в памяти);
- STIME — время старта процесса;
- TTY — указание на терминал, с которого запущен процесс;
- S или STAT — статус процесса;
- PRI — приоритет планирования;
- NI — значение nice (см. описание команды `nice` ниже);
- TIME — сколько времени центрального процессора занял данный процесс;
- CMD или COMMAND — командная строка запуска программы, выполняемой данным процессом;

а также и другие поля, полный список которых приведен на man-странице, посвященной команде `ps`.

Значения, выводимые в большинстве этих полей вы поймете без дополнительных пояснений. В поле **Статус процесса**, как уже говорилось выше, могут стоять следующие значения:

- R — выполнимый процесс, ожидающий только момента, когда планировщик задач выделит ему очередной квант времени;
- S — процесс "спит";
- D — процесс находится в состоянии подкачки на диске;
- T — остановленный процесс;
- Z — процесс-зомби.

Рядом с указателем статуса могут стоять дополнительные символы из следующего набора:

- W — процесс не имеет резидентных страниц;
- < — высоко-приоритетный процесс;
- N — низко-приоритетный процесс;
- L — процесс имеет страницы, заблокированные в памяти.

Вторая группа опций регулирует то, какие именно процессы включаются в вывод команды. Чтобы получить список всех процессов надо использовать команду `ps` с опциями `ax` или `-A`. Вывод в этих двух случаях отличается только в поле CMD: в первом случае выдается полная командная строка запуска программы, а во втором — только имя запущенной программы.

Описание всех опций программы `ps` здесь привести невозможно. Поэтому приведем только несколько примеров ее применения, которые покажут, как пользоваться этой командой в типичных ситуациях.

Для того чтобы увидеть все процессы в системе, используя стандартную форму вывода:

```
[user]$ ps -e
```

Можно к той же команде добавить опцию `-o`, после которой указать через запятую, какие именно поля вы хотите видеть в выводе команды:

```
[user]$ ps -eo pid,user,cmd
```

Для того, чтобы увидеть все процессы в системе, используя форму вывода BSD-систем:

```
[user]$ ps ax
```

Для того, чтобы увидеть все процессы в системе, с применением графического отображения отношения "предок-потомок":

```
[user]$ ps -ef
```

Впрочем, для того, чтобы увидеть "лес" деревьев "предок-потомок", лучше воспользоваться очень интересным аналогом команды `ps -ef` — командой `pstree`.

Для того, чтобы увидеть, сколько % ЦПУ и памяти занимают запущенные вами процессы:

```
[user]$ ps -u
```

Чтобы узнать приоритет процесса и значение `nice`, воспользуйтесь опцией -l:

```
[user]$ ps -l
```

4.2. Команда top

Команда `ps` позволяет сделать как бы "моментальный снимок" процессов, запущенных в системе. В отличие от `ps` команда `top`отображает состояние процессов и их активность "в реальном режиме времени". На рисунке 8.5 изображено окно терминала, в котором запущена программа `top`.

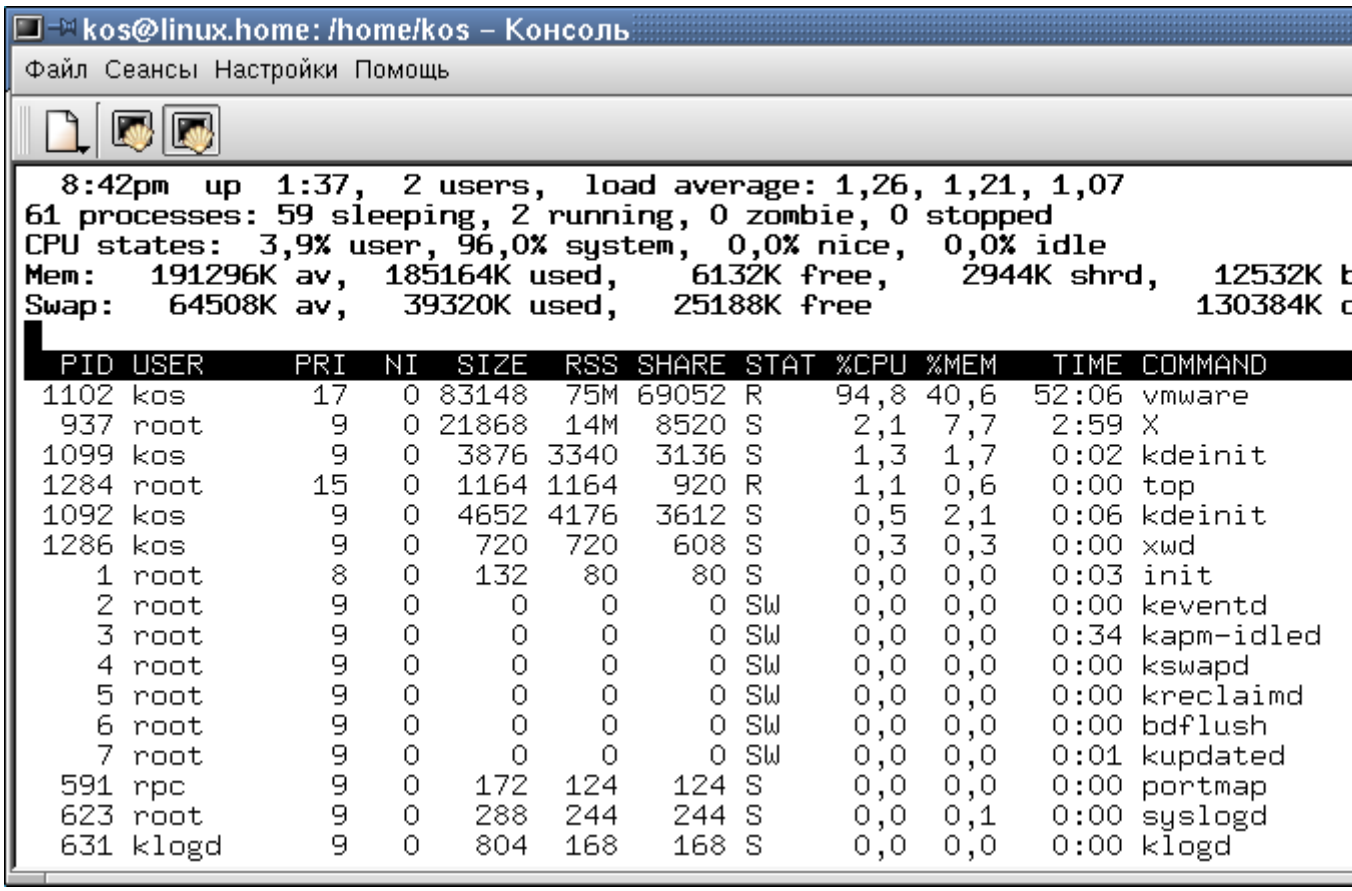

Рис. 1. Вывод команды `top`

Как видите, в верхней части окна отображается астрономическое время, время, прошедшее с момента запуска системы, число пользователей в системе, число запущенных процессов и число процессов, находящихся в разных состояниях, данные об использовании ЦПУ, памяти и свопа. А далее идет таблица, характеризующая отдельные процессы. Число строк, отображаемых в этой таблице, определяется размером окна: сколько строк помещается, столько и выводится. Графы таблицы обозначены так же, как поля вывода команды `ps` *(см.разд. 8.4.1)*, так что дополнительных пояснений здесь не требуется.

Содержимое окна обновляется каждые 5 секунд. Список процессов может быть отсортирован по используемому времени ЦПУ (по умолчанию), по использованию памяти, по PID, по времени исполнения. Переключать режимы отображения можно с помощью команд, которые программа `top` воспринимает. Это следующие команды (просто нажимайте соответствующие клавиши, только с учетом регистра, то есть вместе с клавишей Shift):

- <Shift>+<N> — сортировка по PID;
- <Shift>+<A> — сортировать процессы по возрасту;
- <Shift>+<P> — сортировать процессы по использованию ЦПУ;
- <Shift>+<M> — сортировать процессы по использованию памяти;
- <Shift>+<T> — сортировка по времени выполнения.

Кроме команд, определяющих режим сортировки, команда `top` воспринимает еще ряд команд, которые позволяют управлять процессами в интерактивном режиме. С помощью команды <`K`> можно завершить некоторый процесс (его PID будет запрошен), а с помощью команды <`R`> можно переопределить значение nice для некоторого процесса. Таким образом, эти две команды аналогичны командам `kill` и `renice`,

4.3. Приоритеты, значение nice и команда renice

О том, что такое приоритет, мы уже кратко говорили в начале этой главы, Но некоторые факты надо изложить дополнительно. Приоритет для каждого процесса устанавливается в тот момент, когда процесс порождается. Приоритет процесса определяется так называемым "значением nice", которое лежит в пределах от +20 (наименьший приоритет, процесс выполняется только тогда, когда ничто другое не занимает процессор), до -20 (наивысший приоритет).

Значение nice устанавливается для каждого процесса в момент порождения этого процесса и при обычном запуске команд или программ принимается равным приоритету родительского процесса. Но существует специальная команда `nice`, которая позволяет изменять значение nice при запуске программы. Формат использования этой программы:

```
nice [- adnice] command [args]
```

где `adnice` — значение (от –20 до +19), добавляемое к значению nice процесса-родителя. Полученная сумма и будет значением niceдля запускаемого процесса. 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 niceродительского процесса. Очевидно, что если вы не суперпользователь, то применять эту команду имеет смысл только тогда, когда вы хотите запустить некий процесс с низким значением приоритета.

Другая команда, `renice`, служит для изменения значения nice для уже выполняющихся процессов. Ее формат таков:

```
renice priority [[-p] PID] [[-g] grp] [[-u] user]
```

Например, команда

```
[root]# renice -1 987 –u daemon –p 32
```

увеличивает на 1 приоритет процессов с PID 987 и 32, а также всех процессов пользователя daemon.

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 (увеличить значение nice), но не может увеличить приоритет, даже для возврата значения nice к значению, устанавливаемому по умолчанию. Поэтому процессы с низким приоритетом не могут породить "высокоприоритетных детей".

4.4. Сигналы и команда kill

Сигналы — это средство, с помощью которого процессам можно передать сообщения о некоторых событиях в системе. Сами процессы тоже могут генерировать сигналы, с помощью которых они передают определенные сообщения ядру и другим процессам. С помощью сигналов можно осуществлять такие акции управления процессами, как приостановка процесса, запуск приостановленного процесса, завершение работы процесса. Всего в Linux существует 63 разных сигнала, их перечень можно посмотреть по команде

```
[user]$ kill –l
```

Сигналы принято обозначать номерами или символическими именами. Все имена начинаются на SIG, но эту приставку иногда опускают: например, сигнал с номером 1 обозначают или как SIGHUP, или просто как HUP.

Когда процесс получает сигнал, то возможен один из двух вариантов развития событий. Если для данного сигнала определена подпрограмма обработки, то вызывается эта подпрограмма. В противном случае ядро выполняет от имени процесса действие, определенное по умолчанию для данного сигнала. Вызов подпрограммы обработки называется *перехватом* сигнала. Когда завершается выполнение

подпрограммы обработки, процесс возобновляется с той точки, где был получен сигнал.

Можно заставить процесс игнорировать или блокировать некоторые сигналы. Игнорируемый сигнал просто отбрасывается процессом и не оказывает на него никакого влияния. Блокированный сигнал ставится в очередь на выдачу, но ядро не требует от процесса никаких действий до разблокирования сигнала. После разблокирования сигнала программа его обработки вызывается только один раз, даже если в течение периода блокировки данный сигнал поступал несколько раз.

В табл. 1. приведены некоторые из часто встречающихся сигналов.

Таблица 1. *Сигналы*

№	**Имя**	**Описание**	**Можно перехватывать**	**Можно блокировать**	**Комбинация клавиш**
1	HUP	Hangup. Отбой	Да	Да	
2	INT	Interrupt. В случае выполнения простых команд вызывает прекращение выполнения, в интерактивных программах — прекращение активного процесса	Да	Да	<Ctrl>+<C> или <Del>
3	QUIT	Как правило, сильнее сигнала Interrupt	Да	Да	<Ctrl>+<\>
4	ILL	Illegal Instruction. Центральный процессор столкнулся с незнакомой командой (в большинстве случаев это означает, что допущена программная ошибка). Сигнал отправляется программе, в которой возникла проблема	Да	Да	
8	FPE	Floating Point Exception. Вычислительная ошибка, например, деление на ноль	Да	Да	

9	KILL	Всегда прекращает выполнение процесса	Нет	Нет	
11	SEGV	Segmentation Violation. Доступ к недозволенной области памяти	Да	Да	
13	PIPE	Была предпринята попытка передачи данных с помощью конвейера или очереди FIFO, однако не существует процесса, способного принять эти данные	Да	Да	
15	TERM	Software Termination. Требование закончить процесс (программное завершение)	Да	Да	
17	CHLD	Изменение статуса порожденного процесса	Да	Да	
18	CONT	Продолжение выполнения приостановленного процесса	Да	Да	
19	STOP	Приостановка выполнения процесса	Нет	Нет	
20	TSTR	Сигнал останова, генерируемый клавиатурой. Переводит процесс в фоновый режим	Да	Да	<Ctrl>+<Z>

Как видите, некоторые сигналы можно сгенерировать с помощью определенных комбинаций клавиш. Но такие комбинации существуют не для всех сигналов. Зато имеется команда `kill`, которая позволяет послать заданному процессу любой сигнал. Как уже было сказано, с помощью этой команды можно получить список всех возможных сигналов, если указать опцию `-l`. Если после этой опции указать номер сигнала, то будет выдано его символическое имя, а если указать имя, то получим соответствующий номер.

Для посылки сигнала процессу (или группе процессов) можно воспользоваться командой `kill` в следующем формате:

```
[user]$ kill [-сигн] PID [PID..]
```

где `сигн` – это номер сигнала, причем если указание сигнала опущено, то посылается сигнал 15 (TERM — программное завершение процесса). Чаще всего используется сигнал 9 (KILL), с помощью которого суперпользователь может завершить любой процесс. Но сигнал этот очень "грубый", если можно так выразиться, поэтому его использование может привести к нарушению порядка в системе. Поэтому в большинстве случаев рекомендуется использовать сигналы TERM или QUIT, которые завершают процесс более "мягко".

Естественно, что наиболее часто команду `kill` вынужден применять суперпользователь. Он должен использовать ее для уничтожения процессов-зомби, зависших процессов (они показываются в листинге команды `ps` как `<exiting>`), процессов, которые занимают слишком много процессорного времени или слишком большой объем памяти и т. д. Особый случай — процессы, запущенные злоумышленником. Но обсуждение этого особого случая выходит за рамки данной книги.

4.5. Перевод процесса в фоновый режим

Если вы запускаете какой-то процесс путем запуска программы из командной строки, то обычно процесс запускается, как говорят, "на переднем плане". Это значит, что 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 запуска процесса в фоновом режиме 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. Но прежде, чем рассказывать об этих командах, надо рассказать о команде `jobs`. Она всегда вызывается без аргументов и показывает задания, запущенные из текущего экземпляра shell. В начале каждой строки вывода этой команды указывается порядковый номер задания в виде числа в квадратных скобках. После номера указывается состояние процесса: stopped (остановлен), running (выполняется) или suspended (приостановлен). В конце строки указывается команда, которая исполняется данным процессом. Один из номеров выполняющихся заданий помечен знаком `+`, а еще один — знаком `-`. Процесс, помеченный знаком `+`, будет по умолчанию считаться аргументом команд `fg` или `bg`, если они вызываются без параметров. Процесс, помеченный знаком -, получит знак `+`, если только завершится по какой-либо причине процесс, который был помечен знаком +.

А теперь можно рассказать и о командах `fg` и `bg`, которые служат для перевода процессов на передний план или возврата их в фоновый режим. В качестве аргумента обеим этим командам передаются номера тех заданий, которые присутствуют в выводе команды `jobs`. Если аргументы отсутствуют, то подразумевается задание, помеченное знаком +. Команда `fg` переводит указанный в аргументе процесс на передний план, а команда `bg` — переводит процесс в фоновый режим. Одной командой `bg` можно перевести в фоновый режим сразу несколько процессов, а вот возвращать их на передний план необходимо по одному.

4.6. Команда nohup

Предположим, вы запустили из оболочки `bash` несколько процессов, часть из них в фоновом режиме. И по каким-то причинам завершили текущую сессию работы в оболочке. При завершении сессии оболочка посылает всем порожденным ею процессам сигнал "отбой", по которому некоторые из порожденных ею процессов могут завершиться, что не всегда желательно. Если вы хотите запустить в фоновом режиме программу, которая должна выполняться и после вашего выхода из оболочки, то ее нужно запускать с помощью утилиты `nohup`. Делается это так:

```
nohup команда &
```

Запускаемый таким образом процесс будет игнорировать посылаемые ему сигналы (если это возможно, см. табл. 8.1). Стандартный выходной поток и стандартный поток ошибок при таком запуске команд перенаправляются в файл nohup.out или $HOME/nohup.out.

Команда `nohup` имеет побочный эффект, заключающийся в том, что значение `nice` для запускаемого процесса увеличивается на 5, т. е. процесс выполняется с более низким приоритетом.

Задание на лабораторную работу

Сценарий: Сбор сведений о системе

В данном сценарии изучаются команды, предоставляющие сведения о системе.

Начальные условия: Командная строка после входа в систему.

1. Определить имя текущей UNIX-системы с помощью команды uname -a
2. Вывести содержимое директории proc с помощью команды ls /proc:
3. Вывести текущие пользовательские сеансы с помощью команды who:
4. Вывести список всех примонтированных устройств с помощью команды mount:
5. Вывести загруженность примонтированных дисков с помощью команды df -h:
6. Вывести информацию о всех выполняющихся процессах с помощью команды ps aux:
7. Оставить в выводе предыдущей команды только системные процессы с помощью команды ps aux | grep -v user:
8. Вывести иерархию процессов с помощью команды pstree
9. Рассмотреть поведение процессов интерактивно с помощью команды top.

Сценарий: Управление процессами с помощью сигналов

1. В данном сценарии изучаются сигналы, управляющие клавиши для передачи процессам сигналов, команды для управления процессами.
2. Начальные условия: Командная строка после входа в систему.
3. Запустите команду yes, производящую бесконечный вывод символа y на экран. Прервите её нажатием Ctrl-C.
4. При этом запущенному процессу был отправлен сигнал SIGTERM – завершения программы.
5. Запустим сбор информации обо всех файлах системы с помощью команды find / > files.txt.
6. Найдем идентификатор запущенного только что процесса с помощью команды ps aux | grep find, запущенной в другом терминале.
7. Отправим сигнал завершения этому процессу с помощью команды kill 8178, указав в качестве параметара идентификатор процесса.
8. Если попытаться завершить системный процесс, например командой kill 1, появится сообщение об ошибке доступа:
9. Отправление сигналов системным процессам может производить только суперпользователь.
10. Альтернативным способом отправления сигналов процессам – по имени процесса, а не по PID – является команда killall. Выполнив команду killall bash, мы завершим все командные оболочки, а тем самым и сеансы пользователей.
11. До этого мы отправляли только сигнал завершения процесса. Он может перехватываться и игнорироваться программами. Неперехватываемым является сигнал SIGKILL, который может быть отправлен, например, следующей командой: killall -SIGKILL find

Сценарий: Выполнение задач в фоновом режиме

1. В данном сценарии изучается работа с заданиями командной оболочки, запуск заданий в фоновом режиме.
2. Начальные условия: Командная строка после входа в систему.
3. Запустим длительную команду, например find / > files.txt и приостановим её выполнение с помощью нажатия Ctrl-Z. При этом процессу посылается сигнал SIGSTOP.
4. Команда приостановлена и запомнена как задача 1 (номер в квадратных скобках).
5. Текущий список запущенных задач командной оболочки можно посмотреть командой jobs.
6. Возобновить исполнение задания можно командой fg 1, аргументом которой является номер задания.
7. Если еще раз приостановить процесс, можно запустить задачу в фоновом режиме: bg 1.
8. Тогда можно будет продолжать работу в командной строке.
9. Команды можно сразу запускать в фоновом режиме. При этом необходимо добавить символ «&» (амперсанд) в конец строки команды: find / -name "*.xml" > xml-list &
10. При этом командная оболочка выводит номер задания и PID созданного процесса.

Сценарий: Запуск демонов

1. В данном сценарии рассматриваются демоны как процессы, не связанные ни с одним терминалом.
2. Начальные условия: Командная строка после входа в систему.
3. Если запустить команду в фоновом режиме и выйти из командной оболочки: например, find / -name "*.html" -exec grep -Hn "linux loader" \{} \; & и exit, то запущенная команда завершится по сигналу SIGHUP.
4. Для того, чтобы программы не получали сигнал SIGHUP, используется специальная команда nohup: nohup find / -name "*.txt" -exec grep -Hn "linux loader" \{} \; &
5. Эта программа завершится корректно после окончания поиска.

Сценарий: Изменение приоритетов выполняющихся программ

1. В данном сценарии изучается механизм приоритетов UNIX и команды для изменения приоритетов запускаемых процессов.
2. Начальные условия: Командная строка после входа в систему.
3. Каждый процесс в системе имеет свой уровень приоритета (в UNIX он называется «nice»), который можно увидеть с помощью команды ps -l – в столбце «NI».
4. По умолчанию, приоритет процессов простого пользователя равен нулю.
5. Для запуска процесса с заданным приоритетом воспользуемся командой nice. Напрмер, запустим создание архива с пониженным приоритетом: nice -n 10 tar -cjf libraries.tar.bz2 /usr/lib/.
6. Чтобы изменить приоритет уже запущенной программы используется команда renice. Например, для понижения приоритета заранее запущенного процесса архивации tar -cjf libraries.tar.bz2 /usr/lib & воспользуемся командой renice +10 -p 3442:
7. Простые пользователи не могут повышать приоритет процессов, только понижать.