Лабораторная работа: *Управление процессами в Linux*

Понятие процесса

Каждый раз, когда вы запускаете на выполнение программу, вы начинаете то, что в литературе именуется как процесс. Другими словами – процессом называется выполняемая в данный момент программа или ее потомки. Каждый процесс запускается от имени какого-то пользователя. Процессы, которые стартовали при загрузке, обычно выполняются от имени пользователей root или nobody.

Каждый пользователь может управлять поведением процессов, им запущенных. При этом пользователь root может управлять всеми процессами — как запущенными от его имени, так и процессами, порожденными другими пользователями операционной системы. Управление процессами осуществляется с помощью утилит, а также посредством некоторых команд командной оболочки (shell).

Каждый процесс в системе имеет уникальный номер — идентификационный номер процесса (Process Identification, PID). Этот номер используется ядром операционной системы, а также некоторыми утилитами для управления процессами.

Выполнение процесса на переднем плане и в фоновом режиме

Процессы могут выполняться на переднем плане (foreground) — режим по умолчанию и в фоновом режиме (background). На переднем плане в каждый момент для текущего терминала может выполняться только один процесс. Однако пользователь может перейти в другой виртуальный терминал и запустить на выполнение еще один процесс, а на другом терминале еще один и т. д. Процесс переднего плана – это процесс, с которым вы взаимодействуете, он получает информацию с клавиатуры (стандартный ввод) и посылает результаты на ваш экран (стандартный вывод).

Фоновый процесс после своего запуска благодаря использованию специальной команды командной оболочки отключается от клавиатуры и экрана, т. е. не ожидает ввода данных со стандартного ввода и не выводит информацию на стандартный вывод, а командная оболочка не ожидает окончания запущенного процесса, что позволяет пользователю немедленно запустить еще один процесс.

Обычно фоновые процессы требуют очень большого времени для своего завершения и не требуют вмешательства пользователя во время существования процесса. К примеру, компиляция программ или архивирование большого объема информации — кандидаты номер один для перевода процесса в фоновый режим.

Процессы так же могут быть отложенными. Отложенный процесс — это процесс, который в данный момент не выполняется и временно остановлен. После того как вы остановили процесс, в дальнейшем вы можете его продолжить как на переднем плане, так и в фоновом режиме. Возобновление приостановленного процесса не изменит его состояния — при возобновлении он начнется с того места, на котором был приостановлен.

Для выполнения программы в режиме переднего плана достаточно просто набрать имя программы в командной строке и запустить ее на выполнение.

Для запуска программы в качестве фонового процесса достаточно набрать в командной строке имя программы и в конце добавить знак амперсанта (&), отделенный пробелом от имени программы и ее параметров командной строки, если таковые имеются. Например, команда для запуска программы **yes** в фоновом режиме с подавлением вывода имеет вид:

```
/home/user$ yes > /dev/null &
```

После команды выводится сообщение, состоящее из двух чисел. Первое число в скобках означает номер запущенного фонового процесса для пользователя в текущем сеансе, с его помощью можно производить манипуляции с этим фоновым процессом. Второе число показывает идентификационный номер (PID) процесса. Отличия этих двух чисел достаточно существенные. Номер фонового процесса уникален только для пользователя, запускающего данный фоновый процесс. То есть, если в системе три пользователя решили запустить фоновый процесс (первый для текущего сеанса), то в результате у каждого пользователя появится

фоновый процесс с номером 1. Напротив, идентификационный номер процесса (PID) уникален для всей операционной системы и однозначно идентифицирует в ней каждый процесс. Спрашивается, для чего тогда вводить нумерацию фонового процесса для пользователя? Для удобства. Номер фонового процесса хранится в переменных командной оболочки пользователя и позволяет не забивать голову цифрами типа 2693 или 1294, а использовать переменные типа 1 и т. д.

Для проверки состояния фоновых процессов можно воспользоваться командой командной оболочки — **jobs**.

```
/home/user$ jobs
[1]+   Running          yes >/dev/null &
/home/user$
```

Из вышеприведенного примера видно, что у пользователя **user** в данный момент запущен один фоновый процесс, и он выполняется.

Обратите внимание, что подавление вывода команды осуществляется перенаправлением выходного потока на псевдоустройство /dev/null . Все, что записывается в этот файл, "исчезает" навсегда.

Остановка и возобновление процесса

Помимо прямого указания выполнять программу в фоновом режиме, существует еще один способ перевести процесс в фоновый режим. Для этого мы должны выполнить следующие действия:

1. Запустить процесс выполняться на переднем плане.
2. Приостановить выполнение процесса.
3. Продолжить процесс в фоновом режиме.

Для выполнения программы введем ее имя в командной строке и запустим на выполнение. Для остановки выполнения программы необходимо нажать на клавиатуре следующую комбинацию клавиш — <Ctrl>+<Z>. После этого вы увидите на экране примерно следующее:

```
/home/user$ yes > /dev/null
ctrl+Z
[1]+   Stopped          yes  >/dev/null
/home/user$
```

Для того чтобы перевести выполнение этого процесса в фоновый режим, необходимо выполнить следующую команду:

```
/home/user$ bg  %1
```

Причем необязательно делать это сразу после остановки процесса, главное правильно указать номер остановленного процесса.

Для того чтобы вернуть процесс из фонового режима выполнения на передний план, достаточно выполнить следующую команду:

```
/home/user$ fg %1
```

В том случае, если вы хотите перевести программу в фоновый или, наоборот, на передний план выполнения сразу после остановки процесса, можно выполнить соответствующую программу без указания номера остановленного процесса.

Существует большая разница между фоновым и остановленным процессом. Остановленный процесс не выполняется и не потребляет ресурсы процесса, однако занимает оперативную память или пространство свопинга. В фоновом же режиме процесс продолжает выполняться.

Завершение работы процесса

Рассмотрим возможные способы завершения процесса.

Вариант первый. Если процесс интерактивный, как правило, в документации или прямо на экране написано, как корректно завершить программу.

Вариант второй. В том случае, если вы не знаете, как завершить текущий процесс (не фоновый), можно воспользоваться клавиатурной комбинацией <Ctrl>+<C>. Попробуйте также комбинацию клавиш <Ctrl>+<Break>. А для остановки фонового процесса можно перевести его на передний план, а затем уже воспользоваться вышеприведенными клавиатурными комбинациями.

Вариант третий и самый действенный. В том случае, если вам не удалось прекратить выполнение процесса вышеприведенными способами – например, программа зависла – для завершения процесса можно воспользоваться следующими командами: **kill, killall**.

Команда **kill** может получать в качестве аргумента, как номер процесса, так и идентификационный номер (PID) процесса. Таким образом, команда:

```
/home/user$  kill 123
```

эквивалентна команде:

```
/home/user$  kill %1
```

Можно видеть, что не надо использовать "%", когда вы обращаетесь к работе по идентификационному номеру (PID) процесса.

С помощью команды **killall** можно прекратить выполнение нескольких процессов сразу, имеющих одно и то же имя. Например, команда

```
/home/user$ killall mc
```

прекратит работу всех программ **mc**, запущенных от имени данного пользователя.

Для того чтобы завершить работу процесса, вам надо быть его владельцем. Это сделано в целях безопасности. Если бы одни пользователи могли завершать процессы других пользователей, открылась бы возможность исполнения в системе множества злонамеренных действий. Пользователь root может завершить работу любого процесса в операционной системе.

Программы, используемые для управления процессами

Существует достаточно большое количество утилит, используемых для управления тем или иным способом процессами, исполняемыми в операционной системе. Здесь мы рассмотрим только основные такие утилиты. В табл. 1 приведен список основных программ, тем или иным образом предназначенных для управления процессами.

Табл.1. Программы управления процессами

Программа	**Описание**
at	Выполняет команды в определенное время
batch	Выполняет команды тогда, когда это позволяет загрузка системы
cron	Выполняет команды по заранее заданному расписанию
crontab	Позволяет работать с файлами crontab отдельных пользователей
kill	Прекращает выполнение процесса
nice	Изменяет приоритет процесса перед его запуском
nohup	Позволяет работать процессу после выхода пользователя из системы
ps	Выводит информацию о процессах
renice	Изменяет приоритет работающего процесса
w	Показывает, кто в настоящий момент работает в системе и с какими программами

nohup

Эта утилита позволяет организовать фоновый процесс, продолжающий свою работу даже тогда, когда пользователь отключился от терминала, в отличие от команды &, которая этого не позволяет. Для организации такого фонового процесса необходимо выполнить команду в форме:

nohup ***выполняемая_фоновая_команда &***

Во вновь запущенном терминале процесс нельзя увидеть с помощью команды **jobs**, так как эта команда выводит список процессов текущего терминала, поэтому после подключения к терминалу необходимо использовать команду **ps** с ключом -A.

ps

Программа предназначена для получения информации о существующих в операционной системе процессах. У этой команды есть множество различных опций, но мы остановимся на самых часто используемых. Для получения подробной информации смотрите man-страницу этой программы.

Простой запуск **ps** без параметров выдаст список программ, выполняемых на текущем терминале. Обычно этот список очень мал, например:

```
PID TTY   TIME     CMD
885  tty100:00:00 login
893  tty100:00:00 bash
955  tty100:00:00 ps
```

Первый столбец — PID (идентификационный номер процесса). Как уже упоминалось, каждый выполняющийся процесс в системе получает уникальный идентификатор, с помощью которого производится управление процессом. Каждому вновь запускаемому на выполнение процессу присваивается следующий свободный PID. Когда процесс завершается, его номер освобождается. Когда достигнут максимальный PID, следующий свободный номер будет взят из наименьшего освобожденного.

Следующий столбец — TTY — показывает, на каком терминале процесс выполняется.

Столбец TIME показывает, сколько процессорного времени выполняется процесс. Оно не является фактическим временем с момента запуска процесса, поскольку Linux — это многозадачная операционная система. Показывается время, реально потраченное процессором на выполнение процесса.

Столбец CMD показывает имя программы, опции командной строки не выводятся.

Для получения расширенного списка процессов, выполняемых в системе, используется следующая команда:

```
ps -ax
```

```
PID    TTY   STA   TIM COMMAND
               T     E
  1      ?     S   0:0 init
                     4
  …
437      ?     S   0:0 syslogd -m 0
                     0
442      ?     S   0:0 klogd -2
                     0
 885   tty1    S   0:0 login - -
                     0 root
1067   pts/    R   0:0 ps -ax
          0          0
```

Как правило, список запущенных процессов в системе велик и достаточно сильно зависит от конфигурации операционной системы. Опции, заданные программе в этом примере, заставляют ее выводить не только имена программ, но и список опций, с которыми были запущены программы.

Появился новый столбец — STAT, в котором отображается состояние (status) процесса. Полный список состояний вы можете прочитать в описании программы **ps**. Опишем самые важные состояния:

– буква R обозначает запущенный процесс, исполняющийся в данный момент времени;

– буква S обозначает спящий (sleeping) процесс — процесс ожидает какое-то событие, необходимое для его активизации;

– буква Z используется для обозначения "зомбированных" процессов (zombied) — это процессы, родительский процесс которых прекратил свое существование, оставив дочерние процессы рабочими.

Обратите внимание на колонку TTY. Как вы, наверное, заметили, многие процессы, расположенные в верхней части таблицы, в этой колонке содержат знак "?" вместо терминала. Так обозначаются процессы, запущенные с более не активного терминала. Как правило, это всякие системные сервисы.

Если вы хотите увидеть еще больше информации о выполняемых процессах, попробуйте выполнить команду:

ps –aux

```
USER        PID  %CPU  %MEM    VSZ   RSS TTY      STAT  START  TIME  COMMAND
root          1   1.2   0.2   1412   520 ?        S     14:51  0:04  init
root          o   0.0   0.0      0     0 ?        SW    14:51  0:00  [keventd]
root          3   0.0   0.0      0     0 ?        SW    14:51  0:00  [карт-idled]
root          4   0.0   0.0      0     0 ?        SWN   14:51  0:00  [ksoftirqd_CPUO]
root          5   0.0   0.0      0     0 ?        SW    14:51  0:00  [kswapd]
root          6   0.0   0.0      0     0 ?        SW    14:51  0:00  [kreclaimd]
root          7   0.0   0.0      0     0 ?        SW    14:51  0:00  [bdflush]
root          8   0.0   0.0      0     0 ?        SW    14:51  0:00  [kupdated]
root          9   0.0   0.0      0     0 ?        SW<   14:51  0:00  [mdrecoveryd]
root         13   0.0   0.0      0     0 ?        SW    14:51  0:00  [kjournald]
root        437   0.0   0.2   1472   592 ?        s     14:52  0:00  syslogd -m 0
root        442   0.0   0.4   1928  1040 ?        s     14:52  0:00  klogd -2
rpc       4 62    0.0   0.2   1552   588 ?        s     14:52  0:00  portmap
rpcuser     490   0.0   0.2   1596   756 ?        s     14:52  0:00  rpc.statd
root        590   0.0   0.2   1396   524 ?        s     14:52  0:00  /usr/sbin/apmd -p
root        647   0.0   0.4   2676  1268 ?        s     14:52  0:00  /usr/sbin/sshd
root        680   0.0   0.3   2264   992 ?        s     14:52  0:00  xinetd -stayalive
Ip          704   0.0   0.3   2600  1020 ?        s     14:52  0:00  lpd Waiting
root        732   0.0   0.7   5296  1984 ?        s     14:52  0:00  sendmail: accepti
root        751   0.0   0.1   1440   492 ?        s     14:52  0:00  gpm -t ps/2 -m /d
root        769   0.0   0.2   1584   660 ?        s     14:52  0:00  crond
xfs         835   0.0   1.4   4988  3612 ?        s     14:52  0:00  xfs -droppriv -da
root        853   0.0   0.2   1416   600 ?        s     14:52  0:00  anacron
daemon      871   0.0   0.2   1444   568 ?        s     14:52  0:00  /usr/sbin/atd
root        885   0.0   0.4   2320  1076 ttyl     s     14:52  0:00  login — root
root        886   0.0   0.1   1384   448 tty2     s     14:52  0:00  /sbin/mingetty tt
root        887   0.0   0.1   1384   448 tty3     s     14:52  0:00  /sbin/mingetty tt
root        893   0.0   0.5   2464  1312 ttyl     s     14:52  0:00  -bash
root       1037   0.0   0.7   3284  1804 ttyl     R     14:56  0:00  /usr/bin/mc -P
root       1038   0.0   0.1   1380   348 ?        s     14:56  0:00  cons.saver /dev/t
root       1039   0.0   0.5   2552  1392 pts/0    s     14:56  0:00  bash -rcfile .bas
root       1068   0.0   0.3   2780   824 pts/0    R     14:57  0:00  ps -aux
```

Появились еще столбцы:

- USER — показывает, от имени какого пользователя был запущен данный процесс;
- %CPU, %MEM — показывают, сколько данный процесс занимает соответственно процессорного времени и объем используемой оперативной памяти;
- TIME — время запуска программы.

В табл. 2 приведены некоторые ключи программы **ps**.

Таблица 2. Опции команды ps

Ключ	Описание
a	Показать процессы всех пользователей
c	Имя команды из переменной среды
e	Показать окружение
f	Показать процессы и подпроцессы

h	Вывод без заголовка
j	Формат заданий
l	"Длинный" формат вывода
m	Вывод информации о памяти
n	Числовой вывод информации
r	Только работающие процессы
s	Формат сигналов
S	Добавить время использования процессора порожденными процессами
txx	Только процессы, связанные с терминалом xx
u	Формат вывода с указанием пользователя
v	Формат виртуальной памяти
w	Вывод без обрезки информации для размещения в одной строке
x	Показать процессы без контролирующего терминала

top

Еще одна утилита, с помощью которой можно получать информацию о запущенных в операционной системе процессах. Для использования достаточно просто запустить команду **top** на выполнение. Эта утилита выводит на экран список процессов в системе, отсортированных в порядке убывания значений использованного процессорного времени. Например:

```
2:55pm up 3 min,  1 user,  load average: 0,06, 0,09, 0,03
32 processes: 31 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  1,1% user,  2,9% system,  0,0% nice, 95,8% idle
Mem:   255532K av,   42856K used,  212676K free,    OK shrd,   8560K buff
Swap:  257000K av,      OK used,  257000K free                 19920K cached

  PID USER     PRI  N1  SIZE  RSS  SHARE STAT %CPU %MEM   TIME COMMAND
    1 root       8   0   520  520    452 S     0,0  0,2   0:04 init
    2 root       9   0     0    0      0 SW    0,0  0,0   0:00 keventd
    3 root       9   0     0    0      0 SW    0,0  0,0   0:00 kapm-idled
    4 root      19  19     0    0      0 SWN   0,0  0,0   0:00 ksoftirqd_CPU
                                                                O
    5 root       9   0     0    0      0 SW    0,0  0,0   0:00 kswapd
    6 root       9   0     0    0      0 SW    0,0  0,0   0:00 kreclaimd
    7 root       9   0     0    0      0 SW    0,0  0,0   0:00 bdflush
    8 root       9   0     0    0      0 SW    0,0  0,0   0:00 kupdated
    9 root      -1 -20     0    0      0 SW<   0,0  0,0   0:00 mdrecoveryd
   13 root       9   0     0    0      0 SW    0,0  0,0   0:00 kjournald
  437 root       9   0   592  592    496 S     0,0  0,2   0:00 syslogd
  442 root       9   0  1040  104    448 S     0,0  0,4   0:00 klogd
                                     0
  462 rpc        9   0   588  588    504 S     0,0  0,2   0:00 portmap
  490 rpcuse     9   0   756  756    660 S     0,0  0,2   0:00 rpc.statd
      r
  590 root       8   0   524  524    464 S     0,0  0,2   0:00 apmd
  647 root       9   0  1268 1268   1076 S     0,0  0,4   0:00 sshd
  680 root -     9   0  1008  992    816 S     0,0  0,3   0:00 xinetd
```

Сначала идет общесистемная информация — из нее можно узнать время запуска операционной системы, время работы операционной системы от момента последнего перезапуска системы, количество зарегистрированных в данный момент в операционной системе пользователей, а также минимальную, максимальную и среднюю загрузку операционной системы. Помимо этого, отображается общее количество процессов и их состояние, сколько процентов ресурсов системы используют пользовательские процессы и системные процессы, использование оперативной памяти и свопа.

Далее идет таблица, во многом напоминающая вывод программы **ps**. Идентификационный номер процесса, имя пользователя — владельца процесса, приоритет процесса, размер процесса, его состояние, используемые процессом оперативная память и ресурс центрального процесса, время выполнения и, наконец, имя процесса.

Утилита **top** после запуска периодически обновляет информацию о состоянии процессов в операционной системе, что позволяет нам динамически получать информацию о загрузке системы.

Утилита **top** полностью управляется с клавиатуры. Вы можете получить справку, нажав клавишу **h.** Вот еще несколько полезных команд:

k — используется для отправки сигнала процессу;

u — используется для вывода процессов указанного пользователя;

i — используется для вывода только работающих процессов;

r — используется для изменения приоритета выбранного процесса.

Можно переключать режимы отображения информации с помощью следующих команд:

<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.

kill

Программа kill (в переводе с английского — убить) предназначена для посылки соответствующих сигналов указанному нами процессу. Как правило, это бывает тогда, когда некоторые процессы начинают вести себя неадекватно. Наиболее часто программа применяется, чтобы прекратить выполнение процессов.

Для того чтобы прекратить работу процесса, необходимо знать PID процесса либо его имя. Например, чтобы "убить" процесс 123, достаточно выполнить следующую команду:

```
kill  123
```

В этом случае по умолчанию процессу посылается сигнал SIGTERM (terminate, завершиться). Процесс, получивший данный сигнал, должен корректно завершить свою работу (закрыть используемые файлы, сбросить буферы ввода/вывода и т. п.).

Как обычно, чтобы прекратить работу процесса, вам необходимо быть его владельцем. Само собой, пользователь root может прекратить работу любого процесса в системе.

Иногда стандартное выполнение программы **kill** не справляется с поставленной задачей. Обычно это объясняется тем, что данный процесс завис, либо выполняет операцию, которую с его точки зрения нельзя прервать немедленно. Для прерывания этого процесса можно воспользоваться следующей командой:

```
kill -9  123
```

Ключ -9 указывает посылать процессу другой тип сигнала — SIGKILL. Это приводит к тому, что процесс не производит корректного завершения, а немедленно прекращает свою жизнедеятельность. Помимо этих сигналов, в вашем распоряжении целый набор различных сигналов. Полный список сигналов можно получить, выполнив следующую команду:

```
kill -l
```

Вы увидите внушительный список сигналов:

```
 1) SIGHUP   2) SIGINT    3) SIGQUIT   4)     SIGILL
 5) SIGTRAP  6) SIG7ABRT  7) SIGBUS    8)     SIGFPE
```

9) SIGKILL 10) SIGUSR1 11) SIGSEGV 12) SIGUSR2
13) SIGPIPE 14) SIGALRM 15) SIGTERM и т. д.

killall

Еще один вариант программы **kill**. Используется для того, чтобы завершить работу процессов, носящих одно и то же имя. К примеру, в нашей системе запущено несколько программ файлового менеджера **mc** . Для того чтобы одновременно завершить работу этих программ, достаточно выполнить команду:

```
killall mc
```

Конечно, этим не ограничивается использование данной команды. С ее помощью можно отсылать сигналы группе одноименных процессов. Для получения более подробной информации по этой команде обращайтесь к ее man-странице.

nice

В операционной системе Linux у каждого процесса есть свой приоритет исполнения. Поскольку операционная система многозадачная, то для выполнения каждого процесса выделяется определенное количество времени в соответствии с приоритетом.

Программа **nice** позволяет запустить команду с предопределенным приоритетом выполнения, который задается в командной строке. При обычном запуске все задачи имеют один и тот же приоритет, и операционная система равномерно распределяет между ними процессорное время. Однако с помощью утилиты **nice** можно понизить приоритет какой-либо задачи, таким образом, предоставляя другим процессам больше процессорного времени. Повысить приоритет той или иной задачи имеет право только пользователь root. Синтаксис использования nice следующий:

```
nice -number command
```

Уровень приоритета процесса определяется параметром *number,* при этом большее его значение означает меньший приоритет процесса. Значение по умолчанию — 10, и *number* представляет собой число, на которое должен быть уменьшен приоритет.

К примеру, процесс **top** имеет приоритет, равный -5. Для того чтобы понизить приоритет выполнения процесса на десять, мы должны выполнить следующую команду:

```
nice  10 top
```

В результате процесс **top** получит приоритет, равный 5.

Только пользователь root может поднять приоритет того или иного процесса, используя для этого *отрицательное* значение параметра *number.*

renice

Программа **renice**, в отличие от программы **nice**, позволяет изменить приоритет уже работающего процесса. Формат запуска программы следующий:

```
renice -number PID
```

В общем, программа **renice** работает точно так же, как и nice. Уровень приоритета процесса определяется параметром *number,* при этом большее его значение означает меньший приоритет процесса. Значение по умолчанию — 10, и *number* представляет собой число, на которое должен быть уменьшен приоритет процесса.

Только пользователь root может поднять приоритет того или иного процесса, используя для этого *отрицательное* значение параметра *number*.

at

Одна из основных задач автоматизации администрирования операционной системы — выполнение программ в заданное время или с заданной периодичностью.

Для запуска одной или более команд в заранее определенное время используется команда **at**. В этой команде вы можете определить время и дату запуска той или иной команды. Команда **at** требует, по меньшей мере, двух параметров — время выполнения программы и запускаемую программу с ее параметрами запуска.

Приведенный ниже пример запустит команду **ls** на выполнение в 1 час 5 минут. Каждая команда записывается на отдельной строке, т. е. завершается нажатием клавиши <Enter>, а по окончании ввода всех команд — <Ctrl>+<D> .

```
at  1:05
ls > file
```

Помимо времени, в команде **at** может быть также определена и дата запуска программы на выполнение.

batch

Команда **batch** в принципе аналогична команде **at**. Более того, **batch** представляет собой псевдоним команды **at -b**. Для чего необходима эта команда? Представьте, вы хотите запустить резервное копирование вечером. Однако в это время система очень занята, и выполнение резервирования системы практически парализует ее работу. Для этого и существует команда **batch** — ее использование позволяет операционной системе самой решить, когда наступает подходящий момент для запуска задачи в то время, когда система не сильно загружена.

Формат команды **batch** представляет собой просто список команд для выполнения, следующих в строках за командой; заканчивается список комбинацией клавиш <Ctrl>+<D>. Можно также поместить список команд в файл и перенаправить его на стандартный ввод команды **batch**.

cron и crontab

cron — это программа, выполняющая задания по расписанию, но, в отличие от команды **at**, она позволяет выполнять задания неоднократно. Вы определяете времена и даты, когда должна запускаться та или иная программа. Времена и даты могут определяться в минутах, часах, днях месяца, месяцах года и днях недели.

Программа **cron** запускается один раз при загрузке системы. При запуске **cron** проверяет очередь заданий **at** и задания пользователей в файлах crontab. Если для запуска не было найдено заданий — следующую проверку **cron** произведет через минуту.

Для создания списка задач для программы **cron** используется команда **crontab**. Для каждого пользователя с помощью этой команды создается его собственный crontab-файл со списком заданий, имеющий то же имя, что и имя пользователя.

Каждая строка в файле crontab содержит шаблон времени и команду. Команда выполняется тогда, когда текущее время соответствует приведенному шаблону. Шаблон времени состоит из пяти частей, разделенных пробелами или символами табуляции, и имеет вид:

минуты часы день_месяца месяц день_недели

Эти поля обязательно должны присутствовать в файле. Для того чтобы программа **cron** игнорировала какое-то поле шаблона времени, поставьте в нем символ звездочки (*).

Например, шаблон 10 01 01 * * говорит о том, что команда должна быть запущена в десять минут второго каждого первого числа любого (*) месяца, каким бы днем недели оно ни было. В табл. 3 приведено описание полей cron-файла.

Табл. 3. Параметры таблицы заданий программы cron

Поле	**Описание**
минуты	Указывает минуты в течение часа. Значения от 0 до 59
часы	Указывает час запуска задания. Значения от 0 до 23, где 0 — полночь
день_месяца	Указывает день месяца, в который должна исполняться команда
месяц	Указывает месяц, в который необходимо запускать задание. Значения лежат в пределах от 1 до 12, где 1 — январь
день_недели	Указывает день недели — или как цифровое значение

	от 0 до 7 (0 и 7 означают воскресенье), или используя первые три буквы, например Mon
задание	Командная строка для запуска задания

Ниже приведены несколько примеров команд, исполняемых программой **cron**:

`01 * * * * /usr/bin/script` – команда запускается в 1 минуту каждого часа

`20 8 * * * /usr/bin/script` – команда запускается каждый день в 8:20

`00 6 * * 0 /usr/bin/script` – команда запускается в 6 часов каждое воскресенье

`40 7 1 * * /usr/bin/script` – команда запускается в 7:40 каждое первое число месяца

Для создания и редактирования файла заданий для программы **cron** используется команда **crontab**. Прямое редактирование файла заданий не допускается.

Команда **crontab** имеет следующие параметры командной строки:

- -e — позволяет редактировать компоненты файла (при этом вызывается редактор, определенный в переменной EDITOR);
- -r — удаляет текущий crontab-файл из каталога;
- -l — используется для вывода списка текущих заданий.

Общение между процессами

Процессы могут общаться между собой при помощи каналов, сокетов, разделяемой памяти и т. д. Рассмотрим только один вариант: каналы.

Именованные каналы (FIFO: First In First Out). Данный вид канала создаётся с помощью mknod или mkfifo, и два различных процесса могут обратиться к нему по имени. Пример работы с fifo:

в первом терминале (создаем именованный канал в виде файла pipe и из канала направляем данные с помощью конвейера в архиватор):

```
[root@proxy 1]# mkfifo pipe
[root@proxy 1]# ls -l
total 0
prw-r--r-- 1 root root 0 Nov  9 19:41 pipe
[root@proxy 1]# gzip -9 -c < pipe > out
```

во втором терминале (отправляем в именованный канал данные):

```
[root@proxy 1]# cat /path/to/file > pipe
```

в результате это приведет к сжатию передаваемых данных gzip-ом