Министерство образования Республики Беларусь

ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра технологий программирования

Методические указания для выполнения лабораторной работы № 5 по курсу «Операционные системы и системное программирование»

«Процессы в ОС UNIX: приоритет процессов, выполнение процессов за заданное время»

Полоцк, 2019

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 да 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 ин формацию 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. После этого вы можете работать с программой.

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. Затем программа запускается на выполнение. В отличие от запуска программы в режиме переднего плана мы получим приблизительно следующее сообщение:

```
/home/student# yes > /dev/null &;
[1] 123
```

```
/home/student#
```

Оно состоит из двух чисел и приглашения командной строки. Таким образом, мы запустили программу выполняться в фоновом режиме и получили возможность запустить с той же самой консоли на выполнение еще какую-то программу.

Число [1] означает номер запущенного нами фонового процесса. Как вы узнаете несколько позже, с его помощью можно будет производить манипуляции с нашим фоновым процессом. Значение 123 показывает идентификационный номер (PID) нашего процесса. Отличия этих двух чисел достаточно существенные. Номер фонового процесса уникален только для пользователя, запускающего данный фоновый процесс. То есть если у нас в системе три пользователя решили запустить фоновый процесс (первый для текущего сеанса) — 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 вида %1, %2. Однако допускается пользоваться и идентификационным номером процесса.

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

```
/home/student# jobs
[1]+    Running                                 yes >/dev/null
&
/home/student#
```

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

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:

- Запустить процесс выполняться на переднем плане.
- Остановить выполнение процесса.
- 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 следующее:

```
/home/student#  yes > /dev/null
ctrl+Z
[1]+   Stopped                 yes  >/dev/null
/home/student#
```

Мы получили приглашение командной строки. Для того чтобы перевести выполнение процесса в фоновый режим, необходимо выполнить следующую команду:

```
bg %1
```

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

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

```
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.

Как остановить выполнение фонового процесса? Использование комбинации клавиш <Ctrl>+<Z> не поможет, поскольку процесс находится в фоновом режиме и не реагирует на ввод данных с консоли. Для решения этой проблемы следует переместить процесс на передний план, а затем остановить.

Завершение работы 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 или "слетел" терминал — 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/student#  kill  123
```

эквивалентна команде:

```
/home/student# 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 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.

Таблица 1- Программы управления процессами

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 фонового процесса необходимо выполнить следующую команду:

nohup выполняемая_фоновая_команда &

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

ps

Программа ps 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 терминале. Обычно этот список очень мал:

PID TTY TIME CMD

885 ttyl00:00:00 login

893 ttyl00:00:00 bash

955 ttyl00:00:00 ps

Что означает полученная информация?

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. Запуск команды без параметров ps покажет процессы, выполняемые на текущем терминале.

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

Столбец CMD показывает, что же это за программа. Отображается только имя программы, опции командной строки не выводятся.

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

ps –ax (приведена часть распечатки)

PID TTY	STAT	TIME	COMMAND
1 ?	S	0:04	Init
437 ?	s	0:00	Syslogd –m 0
442 ?	s	0:00	klogd -2
885tty 1	s	0:00	login — root
1067 pts/0	R	0:00	ps –ax

Как можно видеть, 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 этом столбце 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

USE	PI	%CP	%ME	VSZ	RS	TT	STA	STAR	TIM	COMMAND
root	1	1.2	0.2	141	520	9	S	14:51	0:04	init
root	o	0.0	0.0	0	0	9	SW	14:51	0:00	[keventd]
root	3	0.0	0.0	0	0	9	SW	14:51	0:00	[карт-idled]
root	4	0.0	0.0	0	0	7	SWN	14:51	0:00	[ksoftirqd_CPUO
root	5	0.0	0.0	0	0	9	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	9	SW	14:51	0:00	[kupdated]
root	9	0.0	0.0	0	0	9	SW<	14:51	0:00	[mdrecoveryd]
root	13	0.0	0.0	0	0	9	SW	14:51	0:00	[kjournald]
root	437	0.0	0.2	147	592	9	s	14:52	0:00	syslogd -m 0

Как вы видите — информации прибавилось. Появились еще следующие 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 параметры командной строки программы ps.

Таблица 2. Параметры командной строки 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
	Формат сигналов
S	Добавить время использования процессора порожденными сами
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 используемых ресурсов.

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

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	g	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
-j	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

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, отоброжается 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.

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

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.

Что это означает? Вообще-то программа kill предназначена для посылки процессам управляющих сигналов, одним из которых является сигнал sigterm (terminate, завершиться). Этот сигнал посылается процессу при выполнении

программы kill по умолчанию. Процесс, получивший данный сигнал, должен корректно завершить свою работу (закрыть используемые файлы, сбросить буферы ввода/вывода и т. п.). Ключ -9 указывает программе kill 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:

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	17)	SIGCHLD
18)	SIGCONT	19)	SIGSTOP	20	SIGTSTP	21)	SIGTTIN
22)	SIGTTOU	23)	SIGURG	24	SIGXCPU	25)	SIGXFSZ
26	SIGVTALRM	27)	SIGPROF	28	SIGWINC	29)	SIGIO
30)	SIGPWR	31)	SIGSYS	32	SIGRTMI	33)	SIGRTMIN+1
34)	SIGRTMIN+2	35)	SIGRTMIN+3	36	SIGRTMI	37)	SIGRTMIN+5
38)	SIGRTMIN+6	39)	SIGRTMIN+7	40)	SIGRTMIN+8	41)	SIGRTMIN+9
42)	SIGRTMIN+10	43)	SIGRTMIN+11	44)	SIGRTMIN+12	45)	SIGRTMIN+13
46)	SIGRTMIN+14	47)	SIGRTMIN+15	48)	SIGRTMAX-	49)	SIGRTMAX-
50)	SIGRTMAX-	51)	SIGRTMAX-	52)	SIGRTMAX-	53)	SIGRTMAX-
54)	SIGRTMAX-9	55)	SIGRTMAX-8	56)	SIGRTMAX-7	57)	SIGRTMAX-6
58)	SIGRTMAX-5	59)	SIGRTMAX-4	60)	SIGRTMAX-3	61)	SIGRTMAX-2
62)	SIGRTMAX-1	63)	SIGRTMAX				

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 tc. Для того чтобы одновременно завершить работу этих программ, достаточно всего лишь выполнить следующую команду:

killall me

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.

Изменение приоритета выполнения процессов

В операционной системе Linux у каждого процесса есть свой приоритет исполнения. Это очень удобно. Поскольку операционная система многозадачная — то для выполнения каждого процесса выделяется определенное количество времени. Для некоторых задач необходимо выделить побольше, для некоторых можно поменьше. Для этого и предназначен приоритет процесса. Управление приоритетом процесса осуществляется программами nice и renice.

nice

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 имеет приоритет, равный 5.

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

remce

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.

Выполнение процессов в заданное время

Одна из основных задач автоматизации администрирования операционной системы — выполнение программ в заданное время или с заданной периодичностью. Конечно, можно запускать программы самостоятельно, но проводить 24 часа на работе или постоянно удаленно запускать программы в самое неподходящее время (часа в три ночи) — безумие. Для решения этих проблем существует несколько утилит, позволяющих запускать процессы в нужное время.

at

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 на выполнение в 01:01. Для этого введите все, приведенное ниже, с терминала, завершая ввод каждой строки нажатием клавиши <Enter> и по окончании ввода всей команды — <Ctrl>+<D> для ее завершения.

```
at  1:01
Is
echo "Time is  1:01"
```

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

Пользователь root может без ограничения применять практически любые команды. Для обычных пользователей права доступа к команде at определяются файлами /etc/at.allow и /etc/at.deny. В файле /etc/at.allow содержится список тех, кому разрешено использовать команду at, а в файле /etc/at.deny находится список тех, кому ее выполнять запрещено.

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

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 состоит из пяти частей, разделенных пробелами или символами табуляции, и имеет вид:

минуты часы день месяца месяц день недели задание

Первые пять полей представляют собой шаблон времени и обязательно должны присутствовать в файле. Для того чтобы программа cron игнорировала поле шаблона времени, поставьте в нем символ звездочки (*).

Например, шаблон ю 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 таблицы задания cron.

Таблица 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 Моп
задание	Командная строка для запуска задания

Ниже приведены несколько команд, исполняемых программой cron:

– команда запускается в 1 минуту каждого часа:

```
01  *  *  *  *  /usr/bin/script
```

– команда запускается каждый день в 8:20:

```
20  8  *  *  *     /usr/bin/script
```

– команда запускается в 6 часов каждое воскресенье:

```
00  6  *  *  0  /usr/bin/script
```

– команда запускается в 7:40 каждое первое число:

```
40 7  1  *   *  /usr/bin/script
```

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.

Cron также имеет возможность разрешать или запрещать конкретным пользователем свое использование. Для этого существуют файлы /etc/cron.allow и /etc/cron.deny, которые аналогичны описанным ранее /etc/at.allow и /etc/at.deny.

Варианты индивидуальных заданий

№ Варианта	1 задание	2 задание	3 задание	4 задание	5 задание
1	1	5	9	13	14
2	2	6	10	12	15
3	3	7	11	13	16
5	1	8	9	12	14
6	2	4	10	13	15
7	3	5	11	12	16
8	1	6	9	13	14
9	2	7	10	12	15
10	3	8	11	13	16
11	1	4	9	12	14
12	2	5	10	13	15
13	3	6	11	12	16
14	1	7	9	13	14
15	2	8	10	12	15
16	3	4	11	13	16
17	1	5	9	12	14
18	2	6	10	13	15
19	3	7	11	12	16
20	1	8	9	13	14

Задания

1.Запустите программу yes в фоновом режиме с подавлением потока вывода.

2.Запустите программу yes на переднем плане с подавлением потока вывода. Приостановите выполнение программы. Заново запустите программу yes с теми же параметрами, и завершите ее выполнение.

3.Запустите программу yes на переднем плане без подавления потока вывода. Приостановите выполнение программы. Заново запустите программу yes с теми же параметрами, и завершите ее выполнение.

4.Проверьте состояния процессов, воспользовавшись командой jobs.

5.Переведите процесс, который у вас выполняется в фоновом режиме на передний план и остановите его.

6.Переведите любой ваш процесс с подавлением потока вывода в фоновый режим. Проверьте состояния процессов, воспользовавшись командой jobs. Обратите внимание, что процесс стал выполняющемся (Running) в фоновом режиме.

7.Запустите процесс в фоновом режиме, таким образом, чтобы он продолжил свою работу даже после отключения от терминала. Закройте окно и заново запустите консоль. Убедитесь, что процесс продолжил свою работу.

8.Получите информацию о запущенных в операционной системе процессах с помощью утилиты top.

9.«Убейте» два процесса: для одного используйте его PID, а для другого его идентификатор конкретного задания.

10.Попробуйте послать сигнал 1 (SIGHUP) процессу, запущенному с помощью nohup и обычному процессу.

11.Запустите три процесса команды yes. Завершите работу процессов одновременно, используя команду killall.

12.Запустите программу yes в фоновом режиме с подавлением потока вывода. Используя утилиту nice, запустите программу yes с теми же параметрами и с приоритетом, большим на 5. Сравните абсолютные и относительные приоритеты у этих двух процессов.

13.Используя утилиту renice, измените приоритет у одного из потоков yes таким образом, чтобы у обоих потоков приоритеты были равны.

14.Сделайте так, чтобы в xx минут xx часов автоматически выполнилась утилита ls и вывелась строка текста «Lab rab 3. Zadanie 18». Учтите, что вывод будет осуществляться не на экран, а в файл: /var/spool/mail/student (xx минут xx часов – ближайшие несколько минут).

15.Сделайте так, чтобы в xx минут xx часов каждую пятницу автоматически выполнилась утилита ps и вывелась строка текста «Lab rab 3. Zadanie 19». Учтите, что вывод будет осуществляться не на экран, а в файл: /var/spool/mail/student (xx минут xx часов – ближайшие несколько минут).

16.Сделайте так, чтобы в xx минут xx часов каждый xx месяц автоматически выполнилась утилита ps и вывелась строка текста «Lab rab 3. Zadanie 20». Учтите, что вывод будет осуществляться не на экран, а в файл: /var/spool/mail/student (xx минут xx часов – ближайшие несколько минут, xx месяц – текущий месяц).

Вопросы:

1.Объясните, что произойдет, если запустить программу yes в фоновом режиме без подавления потока вывода.

2.Объясните разницу между действием сочетаний клавиш ^Z и ^C.

3.Опишите, что значит каждое поле вывода команды jobs.

4.Назовите главное отличие утилиты top от ps.

5.Почему процесс, запущенный с помощью nohup не «убивается» сигналом 1?