Расширенное администрирование Linux.

Блок 0

v 1.01

Оглавление

Система ведения журналов

Syslog — стандарт отправки сообщений о происходящих в системе событиях (логов), использующийся в компьютерных сетях, работающих по протоколу IP.

Протокол syslog прост: отправитель посылает короткое текстовое сообщение, размером меньше 1024 байт получателю сообщения. Получатель при этом носит имя «syslogd», «syslog daemon», либо же, «syslog server». Сообщения могут отправляться по сети используя TCP-протокол. Как правило, такое сообщение отсылается в открытом виде. Тем не менее, используя специальные средства (такие, как Stunnel, sslio или sslwrap), возможно шифрование сообщений и отправка их по SSL/TLS.

Syslog используется для удобства администрирования и обеспечения информационной безопасности. Он реализован под множество платформ и используется в множестве устройств. Поэтому, использование syslog позволяет обеспечить сбор информации с разных мест и хранение её в едином репозитории.

Командный интерфейс к syslog —logger

История

Syslog был разработан в 1980 году Эриком Оллманом (Eric Allman) как часть проекта Sendmail, и использовался первоначально только для Sendmail. Зарекомендовав себя как стабильное и удобное решение, Syslog был использован и в других приложениях, став стандартом ведения журналов в системах UNIX и GNU/Linux. Позднее появились реализации и под другие операционные системы.

До недавнего времени, syslog использовался как стандарт de facto без каких-либо формальных спецификаций. Поэтому существовало множество реализаций, некоторые из которых были несовместимы друг с другом. Первые шаги по решению этой проблемы были предприняты в 2001 году — протокол syslog был описан в RFC 3164 . Формальная спецификация, стандартизация содержания сообщений и механизм их передачи были выпущены в 2005 году.

Управление типом и подробностью журналируемой информации

Конфигурационный файл syslog.conf

Файл syslog.conf является главным конфигурационным файлом для демона syslogd. Конфигурационный файл syslog.conf представляет собой набор правил. Каждое правило есть - строка, состоящая из селектора и действия, разделенных пробелом или табуляцией. Селектор представляет собой запись в виде источник.приоритет. (источник иногда именуют - категорией) Селектор может состоять из нескольких записей источник.приоритет, разделенных символом ";" . Можно указывать несколько источников в одном селекторе (через запятую). Поле действие - устанавливает журналируемое действие для селектора.

Получив сообщение для записи в журнал (от klogd, от локальной или удаленной программы), syslogd для каждого правила проверяет не подходит ли сообщение под шаблон, определяемый селектором. Если подходит, то выполняется указанное в правиле действие. Для одного сообщения м.б. выполнено произвольное количество действий (т.е. обработка сообщения не прекращается при первом успехе).

Сообщения с уровнем, равным или выше указанного в селекторе, и источником, равным указанному в селекторе, считается подходящим. Звездочка перед точкой соответствует любому источнику, после точки - любому уровню. Слово none после точки - никакому уровню для данного источника. Можно указывать несколько источников в одном селекторе (через запятую).

Источник (он же подсистема) может быть следующим:

- **0 - kern** - Сообщения ядра
- **1 - user** - Сообщения пользовательских программ
- **2 - mail** - Сообщения от почтовой системы.
- **3 - daemon** - Сообщения от тех системных демонов, которые в отличие от FTP или LPR не имеют выделенных специально для них категорий.
- **4 - auth** - Все что связано с авторизацией пользователей, вроде login и su (безопасность/права доступа)
- **5 - syslog** - Система протоколирования может протоколировать сообщения от самой себя.
- **6 - lpr** - Сообщения от системы печати.
- **7 - news** - Сообщения от сервера новостей. (в настоящее время не используется)
- **8 - uucp** - Сообщения от UNIX-to-UNIX Copy Protocol. Это часть истории UNIX и вероятнее всего она вам никогда не понадобится (хотя до сих пор определенная часть почтовых сообщений доставляется через UUCP).
- **9 - cron** - Сообщения от системного планировщика.
- **10 - authpriv** - То же самое, что и auth, однако сообщения этой категории записываются в файл, который могут читать лишь некоторые пользователи (возможно, эта категория выделена потому, что принадлежащие ей сообщения могут содержать открытые пароли пользователей, которые не должны попадать на глаза посторонним

людям, и следовательно файлы протоколов должны иметь соответствующие права доступа).
- **11 - ftp** - При помощи этой категории вы сможете сконфигурировать ваш FTP сервер, что бы он записывал свои действия.
- **12 - NTP** - сообщения сервера времени
- **13 - log audit**
- **14 - log alert**
- **15 - clock daemon** - сообщения демона времени
- **с 16 по 23 local0 - local7** Зарезервированные категории для использования администратором системы. Категория local7 обычно используется для сообщений, генерируемых на этапе загрузки системы.
- **mark** (не имеющая цифрового эквивалента) - присваивается отдельным сообщениям, формируемым самим демоном syslogd

Под приоритет (степени важности) сообщений заданы 8 уровней важности, которые кодируются числами от 0 до 7:

- **0 - emerg** (старое название PANIC) - Чрезвычайная ситуация. Система неработоспособна.
- **1 - alert** - Тревога! Требуется немедленное вмешательство.
- **2 - crit** - Критическая ошибка (критическое состояние).
- **3 - err** (старое название ERROR) - Сообщение об ошибке.
- **4 - warning** (старое название WARN) - Предупреждение.
- **5 - notice** - Информация о каком-то нормальном, но важном событии.
- **6 - info** - Информационное сообщение.
- **7 - debug** - Сообщения, формируемые в процессе отладки.

Согласно действию, указанному в правиле, сообщение может быть записано в следующие назначения:

- **Обычный файл** - Задается полным путем, начиная со слеша (/). Поставьте перед ним дефис (-), чтобы отменить синхронизацию файла после каждой записи. Это может привести к потере информации, но повысить производительность.
- **Именованные каналы** - Размещение перед именем файла символа канала (|) позволит использовать fifo (first in — first out, первый пришел — первый вышел) или именованный канал (named pipe) в качестве приемника для сообщений. Прежде чем запускать (или перезапускать) syslogd, необходимо создать fifo при помощи команды mkfifo. Иногда fifo используются для отладки.
- **Терминал и консоль** - Терминал, такой как /dev/console.
- **Удаленная машина** - Чтобы сообщения пересылались на другой хост, поместите перед именем хоста символ (@). Обратите внимание, что сообщения не пересылаются с принимающего хоста. (для работы данного назначения на клиенте и сервере в файле /etc/services должна быть прописана строчка syslog 514/udp, и открыт UTP-порт 514)

- **Список пользователей** - Разделенный запятыми список пользователей, получающих сообщения (если пользователь зарегистрирован в системе). Сюда часто включается пользователь root.
- **Все зарегистрированные пользователи** - Чтобы известить всех зарегистрированных пользователей при помощи команды wall, используйте символ звездочки (*).

Пример несложного syslog.conf:

```
# Все сообщения ядра выдавать на консоль.
#kern.*                                                 /dev/console

# Все логи уровня info или выше, кроме сообщений электронной почты, а так же
# не логировать сообщения аутентификации и сообщений демона cron!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# Записывать в отдельный файл сообщения, содержащие конфиденциальную
# информацию аутентификации, независимо от их уровня.
authpriv.*                                              /var/log/secure

# Все сообщения почтовой системы тоже записывать в отдельный файл.
mail.*                                                  -/var/log/maillog

# Логировать сообщения планировщика в файл /var/log/cron
cron.*                                                  /var/log/cron

# Сообщения о чрезвычайных ситуациях должны немедленно получить
# все пользователи системы
*.emerg                                                 *

# Сохранять сообщения новостей уровня crit и выше в отдельный файл.
uucp,news.crit                                          /var/log/spooler

# Сохранять сообщения загрузки в boot.log
local7.*                                                /var/log/boot.log
```

Как и во многих конфигурационных файлах, синтаксис следующий:

- Строки, начинающиеся с #, и пустые строки игнорируются.
- Символ * может использоваться для указания всех категорий или всех приоритетов.
- Специальное ключевое слово none указывает, что журналирование для этой категории не должно быть выполнено для этого действия.
- Дефис перед именем файла (как -/var/log/maillog в этом примере) указывает, что после каждой записи журнал не должен синхронизироваться. В случае аварии системы вы можете потерять информацию, но отключение синхронизации позволит повысить производительность.
- В синтаксисе конфигурационного файла можно поставить перед приоритетом знак !, чтобы показать, что действие не должно применяться, начиная с этого уровня и выше. Подобным образом, перед приоритетом можно поставить знак =, чтобы показать, что

правило применяется только к этому уровню, или !=, чтобы показать, что правило применяется ко всем уровням, кроме этого.

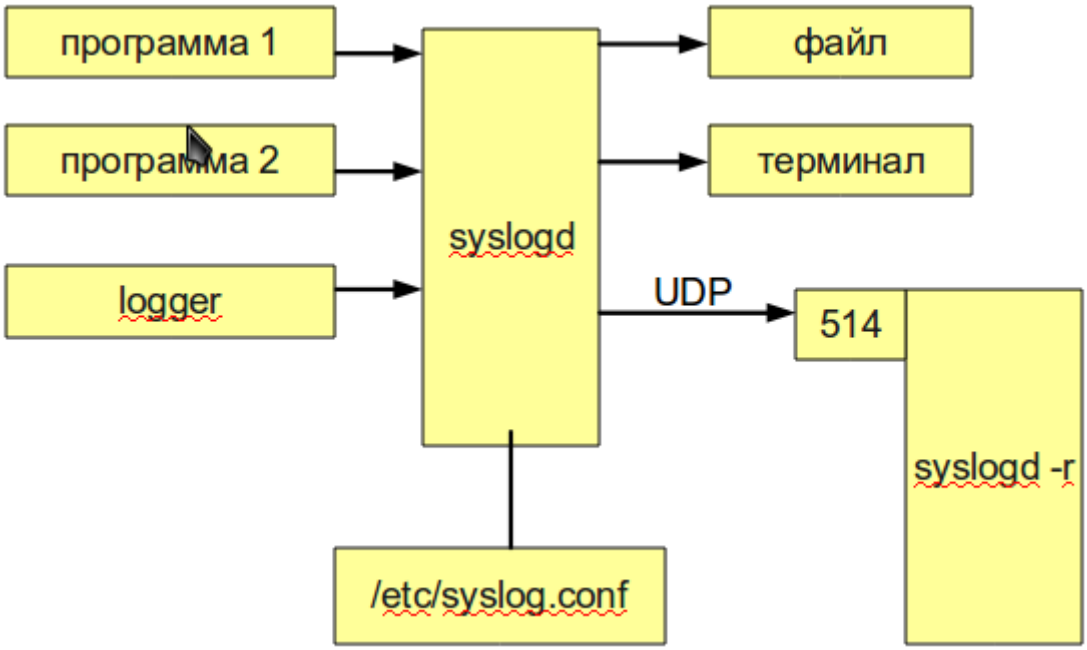
Ниже показано несколько примеров (man syslog.conf можно найти множество других примеров):

```
# Посылать все сообщения ядра в /var/log/kernel.
# Посылать все сообщения уровня critical и higher на удаленную машину sysloger и на консоль
# Посыласть все сообщения уровня  info, notice и warning в /var/log/kernel-info
#
kern.*                          /var/log/kernel
kern.crit                       @sysloger
kern.crit                       /dev/console
kern.info;kern.!err             /var/log/kernel-info
```

Посылать все сообщения почтовой системы, кроме уровня info в /var/log/mail.

```
mail.*;mail.!=info              /var/log/mail
```

Работу syslogd можно показать на схеме:



Вот некоторые возможные параметры запуска демона syslogd:

-a /folder/socket - указание дополнительного слушающего сокета (не забудьте предварительно создать сокет)

-d - отладочный режим. При этом демон не переходит в фоновый режим и выдает все сообщения на текущий терминал;

-f имя-конфигурационного-файла. Задает имя альтернативного конфигурационного файла, который будет использоваться вместо заданного по умолчанию /etc/syslog.conf;

-l список-хостов - задание списка хостов, имена которых не должны записываться с указанием полного доменного имени (FQDN - Full Qwalified Domain Name);

-m минут - запущенный без этой опции sysklogd через каждые 20 минут записывает в протокол сообщения категории mark (временные отметки). С помощью опции -m можно либо изменить интервал между отметками, либо вовсе отменить выдачу таких сообщений;

-p socket - задание альтернативного сокета UNIX (вместо прослушиваемого по умолчанию /dev/log);

-r - разрешение принимать сообщения от удаленных хостов;

-x - запрет определения имени хоста по его адресу для предотвращения зависания при работе на одном хосте с сервером DNS.

-v - показать версию и закончить работу

После запуска демона syslogd создается файл с идентификационным номером процесса /var/run/syslogd.pid.

С помощью команды

```
kill -SIGNAL `cat /var/run/syslogd.pid`
```

можно послать демону syslogd один из следующих сигналов: SIGHUP - перезапуск демона; SIGTERM - завершение работы; SIGUSR1 - включить/выключить режим отладки.

В системе запускаются два демона протоколирования - syslogd и klogd. Оба демона входят в состав пакета sysklogd.

Демон klogd отвечает за журналирование событий, происходящих в ядре системы. Необходимость в отдельном демоне klogd объясняется тем, что ядро не может использовать стандартную функцию syslog. Дело в том, что стандартные библиотеки C (включая ту библиотеку, в которой находится функция syslog) предназначены для использования только обычными приложениями. Поскольку ядро тоже нуждается в функциях журналирования, в него включены свои библиотеки, недоступные приложениям. Поэтому ядро использует свой собственный механизм генерации сообщений.

Демон klogd предназначен для организации обработки этих сообщений. В принципе он может производить такую обработку полностью самостоятельно и независимо от syslogd, например, записывая эти сообщения в файл, но в большинстве случаев используется принятая по умолчанию настройка klogd, при которой все сообщения от ядра пересылаются тому же демону syslogd.

Лабораторная работа

Цель: Научиться настраивать протоколирование событий

Задание:

Отредактируйте файл /etc/default/syslogd изменив строку:

```
SYSLOGD=""
```

на

```
SYSLOGD="-r"
```

В файл /etc/syslog.conf внесите следующие строки:

```
auth.notice                    @IP_адрес_преподавателя
auth.notice                    @IP_адрес_соседа
```

Перезапустите сервер логов командой

```
service sysklogd restart
```

Убедитесь, что сервер логов слушает запросы на 514-м порту по UDP с помощью команды:

```
netstat -nlp4 | grep  :514
```

Создайте скрипт с именем log_write следующего содержания:

```
#!/bin/bash
logger -p auth.notice -t $0 "$@"
```

Сделайте его исполняемым:

```
chmod +x log_write
```

Выполните скрипт:

```
./log_write Test
```

Просмотрите содержимое лог-файла:

```
grep log_write /var/log/auth.log
```

Автоматическая ротация журналов

Со временем, файл журнала имеет свойство увеличиваться, особенно при интенсивной работе какого-либо сервиса. Соответственно, необходимо иметь возможность контролировать размер журналов. Это делается при помощи команды logrotate, которая обычно выполняется демоном cron. О работе cron я расскажу в следующих статьях. Главная цель команды

logrotate состоит в том, чтобы периодически создавать резервные копии журналов и создавать новые чистые журналы. Сохраняется несколько поколений журналов и, когда завершается срок жизни журнала последнего поколения, он может быть заархивирован (сжат). Результат может быть отправлен по почте, например, ответственному за ведение архивов.

Для определения порядка ротации и архивирования журналов используется конфигурационный файл /etc/logrotate.conf. Для разных журналов можно задать разную периодичность, например, ежедневно, еженедельно или ежемесячно, кроме того, можно регулировать количество накапливаемых поколений, а также указать, будут ли копии архивов отправляться ответственному за ведение архивов и, если будут, когда. Ниже показан пример файла /etc/logrotate.conf:

```
# сначала заданы параметры "по-умолчанию" (глобальные опции)
# обновлять файлы журнала еженедельно
weekly

# хранить архив логов за 4 последние недели
rotate 4

# создавать новый (пустой) файл после ротации (обновления)
create

# раскомментируйте, если желаете, чтобы сохраненные файлы сжимались
#compress

# включить настройки ротации из указанного каталога
include /etc/logrotate.d

# не хранить wtmp, или btmp -- настройки ротации данных журналов следующие:
/var/log/wtmp {
 missingok
 monthly
 create 0664 root utmp
 rotate 1
}

/var/log/btmp {
 missingok
 monthly
 create 0660 root utmp
 rotate 1
}
```

Глобальные опции размещаются в начале файла logrotate.conf. Они используются по умолчанию, если где-то в другом месте не задано ничего более определенного. В примере ротация журналов происходит еженедельно и резервные копии сохраняются в течение четырех недель. Как только производится ротация журнала, на месте старого журнала автоматически создается новый. Файл logrotate.conf может содержать спецификации из других файлов. Так, в него включаются все файлы из каталога /etc/logrotate.d.

В этом примере также содержатся специальные правила для /var/log/wtmp и /var/log/btmp (хранящие информацию о удачных и неудачных попытках входя в систему), ротация которых происходит ежемесячно. Если файлы отсутствуют, сообщение об ошибке не выдается. Создается новый файл и сохраняется только одна резервная копия.

В этом примере по достижении резервной копией последнего поколения она удаляется, поскольку не определено, что следует с ней делать.

Резервные копии журналов могут также создаваться, когда журналы достигают определенного размера, и могут быть созданы скрипты из наборов команд для выполнения до или после операции резервного копирования. Пример:

```
/var/log/messages {
 rotate 5
 mail logadmin@sysloger
 size 100k
 postrotate
 /usr/bin/killall -HUP syslogd
 endscript
 }
```

В этом примере ротация /var/log/messages производится по достижении им размера 100 КБ. Накапливается пять резервных копий, и когда истекает срок жизни самой старой резервной копии, она отсылается по почте на адрес logadmin@sysloger. Командное слово postrotate включает скрипт, перезапускающий демон syslogd после завершения ротации путем отправки сигнала HUP. Командное слово endscript необходимо для завершения скрипта, а также в случае, если имеется скрипт prerotate. Более полную информацию см. в страницах руководства man для logrotate.

Параметры, задаваемые в конфигурационном файле logrotate.conf:

- **compress | nocompress** (старые версии сжимаются или не сжимаются с помощью gzip)
- **compresscmd** (задает программу сжатия, по умолчанию - gzip)
- **uncompresscmd** (задает программу разжатия, по умолчанию - ungzip)
- **compressext** (задает суффикс для сжатых файлов)
- **compressoptions** (задает параметры программы сжатия; по умолчанию - "-9", т.е. максимальное сжатие для gzip)
- **copytruncate | nocopytruncate** (обычно старая версия переименовывается и создается новая версия журнала; при задании этого параметра logrotate копирует журнал в новый файл, а затем

обрезает старый; используется, если программа, создающая журнал, не умеет его закрывать; теряются записи, сделанные в промежутке между копированием и обрезанием; а поможет ли, если создающая журнал программа вместо режима append просто пишет в файл, используя внутренний указатель?)

- **create** [права-доступа владелец группа] | **nocreate** (сразу после переименования старой версии журнала и до вызова postrotate создается новый журнал с указанными атрибутами - права доступа задаются в восьмеричном виде, как в chmod.2; если атрибуты не указаны, то берутся от старого журнала)
- **daily** (смена версий в серии происходит ежедневно)
- **delaycompress | nodelaycompress** (некоторые программы не сразу закрывают журнал, в этом случае сжатие надо отложить до следующего цикла)
- **errors email** (кому направлять сообщения об ошибках)
- **extension** суффикс (задается суффикс, добавляемый к именам файлов при ротации перед суффиксом сжатия)
- i**fempty | notifempty** (смена версий даже если файл пуст; действует по умолчанию)
- **include** имя-файла | имя-директории (текстуально подставить файл или все файлы из указанной директории; не включаются поддиректории, специальные файлы и файлы с суффиксами из списка исключений; нельзя использовать внутри секции)
- **mail** адрес | **nomail** (когда смена версий приводит к необходимости удалить старый журнал, то послать его по указанному адресу)
- **mailfirst** (посылать не удаляемую версию журнала, а первую)
- **maillast** (посылать удаляемую версию журнала; действует по умолчанию)
- **missingok | nomissingok** (не посылать сообщения об ошибке, если журнал отсутствует)
- **monthly** (смена версий происходит ежемесячно)
- **olddir** директория | **noolddir** (во время смены версий журнал перемещается в указанную директорию; д.б. на том же физическом устройстве)
- **postrotate** (все дальнейшие строчки до строки endscript исполняются как команды shell после процесса смены версии)
- **prerotate** (все дальнейшие строчки до строки endscript исполняются перед процессом смены версии)
- **rotate** число (сколько старых версий хранить; если 0, то ни одной)

- **size** байт (смена версии происходит, если размер журнала превысил указанное число; можно использовать суффиксы "k" - килобайт - и "M" - мегабайт)
- **sharedscripts | nosharedscripts** (выполнять команды prerotate и postrotate только один раз для всех файлов, описанных в секции)
- **tabooext** [+] список-суффиксов (задание списка суффиксов-исключений для include; если указан знак "плюс", то дополнение, иначе замена; по умолчанию: .rpmorig, .rpmsave, .rpmnew, ",v", .swp и "~")

• **weekly** (смена версий происходит еженедельно)

Лабораторная работа

Цель: Научиться настраивать ротацию журнальных файлов

Задание:

Ознакомьтесь с содержимым /var/messages*

```
ls -l /var/messages*
```

Создайте файл /etc/logrotate.d/messages следующего содержания:

```
/var/log/messages {
compress
size=100
}
```

Выполните ротацию логов:

```
logrotate  /etc/logrotate.conf
```

Выполнение заданий по расписанию

cron — демон-планировщик задач в UNIX-подобных операционных системах, использующийся для периодического выполнения заданий в определённое время. Регулярные действия описываются инструкциями, помещенными в файлы crontab, которые находятся в каталоге[1]:

```
/usr/spool/cron/crontabs
```

или

```
/var/spool/cron/crontabs
```

редактировать их вручную не рекомендуется, для этого используют команду crontab -e.

Основной файл конфигурации cron, /etc/crontab, выглядит примерно так:

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
 17 *  *   *   *  root   cd / && run-parts --report /etc/cron.hourly
 25 6   *    *    *   root    test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.daily )
 47 6   *    *    7   root    test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.weekly )
 52 6   1    *    *   root    test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.monthly )
```

Замена текстового редактора для crontab

По умолчанию crontab использует vi или nano в качестве текстового редактора. Сменить редактор можно командой

```
export EDITOR='mcedit'
```

или

```
export EDITOR='nano'
```

также можно воспользоваться скриптом **select-editor**

Каждый пользователь системы имеет свой файл заданий crontab, в котором описано, в какое время и какие программы запускать от имени этого пользователя. Для редактирования файла crontab используется специальная одноименная программа crontab, позволяющая не прерывать процесс cron на время редактирования.

Для редактирования файла crontab вашего пользователя используется команда:

```
crontab -e
```

Таблица crontab состоит из 6 колонок, разделяемых пробелами или табуляторами. Первые пять колонок задают время выполнения (Минута, Час, День, Месяц, День недели), в них может находиться число, список чисел, разделённых запятыми, диапазон чисел, разделённых тире или символ '*'. Все остальные символы в строке интерпретируются как выполняемая команда с её параметрами. Если команда отправляет какой-нибудь текст в стандартный вывод, этот текст отправляется по e-mail пользователю.

```
* * * * * выполняемая команда
- - - - -
| | | | |
| | | | ----- День недели (0 - 7) (Воскресенье =0 или =7)
| | | ------- Месяц (1 - 12)
| | --------- День (1 - 31)
| ----------- Час (0 - 23)
------------- Минута (0 - 59)
```

Пример файла crontab:

m h dom mon dow command

0 7 * * * /usr/bin/mplayer 2>&1 > /dev/null /home/username/ring.ogg

Лабораторная работа

Цель: Научиться планировать выполнение заданий по расписанию

Задание:

Выполните следующую команду:

```
crontab -e
Select an editor.  To change later, run 'select-editor'.
  1. /bin/ed
  2. /bin/nano        <---- easiest
  3. /usr/bin/emacs23
  4. /usr/bin/mcedit-debian
  5. /usr/bin/vim.basic
  6. /usr/bin/vim.tiny
```

Choose 1-6 [2]: Enter

и отредактируйте файл задания следующим образом:

```
# m h  dom mon dow   command
*/1 0 * * * /bin/date > /tmp/date.log
```

Подождите минуту и просмотрите содержание файла /tmp/date.log

Anacron — анахроничный cron

Anacron в отличие от cron не поддерживает запуск заданий по расписанию, вместо этого задания запускаются с заданым интервалом времени. Это очень удобно для систем которые работают не регулярно, например домашние рабочие станции или ноутбуки. Задачи Anacron хранятся в файле конфигурации /etc/anacrontab:

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# These replace cron's entries

1       5       cron.daily       nice run-parts --report /etc/cron.daily

7       10      cron.weekly      nice run-parts --report /etc/cron.weekly

@monthly        15      cron.monthly nice run-parts --report /etc/cron.monthly
```

Отметки выполнения хранятся в:

```
/var/spool/anacron
```

здесь хранятся файлы расписаний для каждого из пользователей, например root.

Аналогично cron задания определяются набором полей:

```
* * * выполняемая команда

- - -

| | |

| | --------- идентификатор

| ----------- задержка

------------- период
```

Период — период выполнения в днях. Задержка — задержка запуска в минутах. Идентификатор задания — любой непустой символ, кроме / \. Задержка чаще всего используется для того чтобы позволить системе полностью загрузиться.

Однократное выполнение заданий по расписанию

at — unix-утилита, читающая команды со стандартного входного потока и группирующая их в виде задания at для выполнения позже, в заданное время.

Данная команда также доступна в среде MS Windows. Используйте at /? для уточнения параметров для данной ОС.

at и batch читают команды из стандартного ввода или заданного файла которые будут выполнены в определённое время, используя /bin/sh.

- **at** - запускает команды в заданное время.
- **atq** - список заданий, заданных пользователем, если пользователь не суперпользователь; в этом случае, выдаются все задания.
- **atrm** - удаляет задания. batch запускает команды, когда уровни загрузки системы позволяют это делать; в других, когда средняя загрузка системы, читаемая из /proc/loadavg опускается ниже 0.5, или величины, заданной при вызове atrun.

At позволяет некоторые умеренно сложные спецификации времени HHMM (ЧасыМинуты) или HH:MM (Часы:Минуты) для запуска задания в определённое время дня. (Если это время уже прошло, то устанавливается следующий день.) Вы можете также задать midnight(полночь), noon(полдень), или teatime(4 часа после полудня) (4pm), а также задать суффикс времени для AM (до полудня) или PM (после полудня) для запуска утром или вечером. Вы также можете указать, что день, в который будет запущено задание, задаётся датой в форме имя_месяца день год (необязательно), или задать дату в форме MMDDYY, MM/DD/YY или DD.MM.YY. Заданная вами дата должна содержать параметр времени дня. Вы также можете задать время как now + count time-units (текущее время + счётчик временных единиц), где временные единицы могут быть минутами, часами, днями или неделями. Вы можете указать at запустить задание сегодня, используя суффикс времени today, а для запуска задания завтра — суффикс tomorrow.

Суперпользователь может использовать эти команды в любом случае. Для других пользователей, право на использование at определяется файлами /etc/at.allow и /etc/at.deny.

Если файл /etc/at.allow существует, то только пользователи, чьи имена указаны в этом файле могут использовать at.

Если /etc/at.allow не существует, то проверяется /etc/at.deny, каждый пользователь, чьё имя не указано в этом файле может использовать at.

Если не существует ни одного из вышеперечисленных файлов, то только суперпользователю позволяется использовать at

В случае пустого /etc/at.deny считается, что каждому пользователю позволяется использовать эти команды, это установлено по умолчанию.

Лабораторная работа

Цель: Научиться планировать задания средствами at

Задание:

Установите утилиту notify-send командой:

```
sudo apt-get install libnotify-bin
```

Создайте новое задание командой:

```
echo "notify-send 'пора пообедать'" | at now + 1 munute
```

Через минуту вы увидите уведомление