Впервые опубликовано на developerWorks 02.09.2005

Учебник для экзамена LPI 201: Поддержка системы

Администрирование, средний уровень (LPIC-2) тема 211

Журналирование системных сообщений

О журналировании

Многие процессы и серверы под Linux записывают информацию, касающуюся изменения состояния, в так называемые "log файлы" (файлы системных журналов). Эти файлы системного журнала обычно находятся в каталоге /var/log/ и часто начинаются с фиксирования момента времени, указывающего, когда описанное событие произошел. Но как бы там ни было, нет ни условленной последовательности записи, ни определенного формата файлов системного журнала. Одна особенность, на которую вы в значительной степени все-таки можете рассчитывать -- то, что файлы системного журнала Linux являются простыми файлами ASCII, и в каждой строке файла содержится одно "событие". Часто (но не всегда) файлы системного журнала содержат относительно последовательный набор полей или разграниченных таблицей областей данных.

Некоторые процессы, особенно службы интернет, управляют файлом системного журнала, записываемого в пределах их собственного процесса. На самом деле, запись в файл системного журнала –- это, всего лишь, приобщение к открытому файлу дополнительных данных. Но многие программы (особенно демоны и процессы `cron`) используют стандарт syslog API, чтобы позволить управлять процессом журналирования демонам syslogd или klogd.

Анализ файлов системных журналов

Как именно вы будете анализировать то, что записано в log файле, зависит от формата, который используется. Для файлов системного журнала с форматом таблицы, вероятно, будут полезны инструменты типа cut, split, head и tail. grep -- самый мощный инструмент для обнаружения и фильтрации интересующего вас содержания для всех файлов системного журнала. Для более сложных задач обработки вы, вероятно, будете использовать sed, awk, perl или python.

Хорошим введением в область инструментов обработки текста, которые вы должны будете и, скорее всего, будете использовать для обработки и анализа файлов системного журнала, мог бы быть учебник IBM developerWorks Дэвида об утилитах обработки текста GNU. Существует также множество инструментов высокого уровня, чтобы обрабатывать файлы системного журнала, но эти инструменты обычно зависят от дистрибутива и/или не стандартизированы (но часто являются свободными).

Журналирование системных сообщений с помощью syslogd и klogd

Демон klogd перехватывает и журналирует сообщения ядра Linux. Как правило, klogd использует более общие способности syslogd, но в специальных случаях может записывать сообщения непосредственно в файл.

Общий демон syslogd обеспечивает протоколирование для многих программ. Каждое системное сообщение содержит, по крайней мере, поле для времени, поле для имени host'a и обычно поле для имени программы. Поведением syslogd управляет файл конфигурации /etc/syslog.conf. Сообщения от приложений (включая ядро) могут быть зарегистрированы в файлах, которые обычно находятся в /var/log/ или удаленно по сетевому сокету.

Конфигурирование /etc/syslog.conf

Файл /etc/syslog.conf содержит ряд правил, по одному в каждой строчке. Пустые строки и строки, начинающиеся с "*" игнорируются. Каждое правило состоит из двух полей, разделенных пробелом, поля отбора и поля действия. Поле отбора, в свою очередь, содержит одну или более разделенных точкой пар средство - приоритет. Средство - подсистема, которая хотела бы регистрировать свои сообщения и может иметь значения: `auth`, `authpriv`, `cron`, `daemon`, `ftp`, `kern`, `lpr`, `mail`, `mark`, `news`, `security`, `syslog`, `user`, `uucp` и `local0` через `local7`.

Приоритеты имеют определенный порядок, и данному приоритету соответствует значение "этот или выше", если в начале используется "=" (или "! ="). Приоритеты в порядке возрастания: `debug`, `info`, `notice`, `warning` или `warn`, `err` или `error`, `crit`, `alert`, `emerg` или `panic` (несколько имен имеют синонимы). `none` означает, что нет приоритета.

И средства и приоритеты могут принимать значение "*" (wildcard). Несколько средств могут быть разделены запятой, и разные селекторы могут быть разделены точкой с запятой. Например:

```
# from /etc/syslog.conf
# all kernel mesages
kern.*                    -/var/log/kern.log
# `catch-all' logfile
*.=info;*.=notice;*.=warn;\
auth,authpriv.none;\
cron,daemon.none;\
mail,news.none          -/var/log/messages
# Emergencies are sent to everybody logged in
*.emerg                   *
```

Конфигурирование удаленного журналирования системных сообщений

Чтобы сделать возможным удаленное журналирование сообщений syslogd (сообщения реальных приложений, но обработанные syslogd), вы должны сначала разрешить службе "syslog" прослушивание машин и пересылку. Для этого нужно добавить следующую строчку к каждому файлу конфигурации /etc/services:

```
syslog    514/UDP
```

Чтобы настроить локальный syslogd для отправления сообщений удаленному хосту, вы определяете обычное средство и приоритет, но при обозначении действия начинаете с символа "@" для адреса хоста. Хост можно сконфигурировать обычным способом либо в файле /etc/hosts либо через DNS (имя хоста не обязательно должно быть разрешено через resolving, когда syslogd запускается впервые). Например:

```
# from /etc/syslog.conf
# log all critical messages to master.example.com
*.crit                @master.example.com
# log all mail messages except info level to mail.example.com
mail.*;mail.!=info    @mail.example.com
```

Ротация файлов системных сообщений

Вряд ли вы захотите, чтобы ваши файлы системных журналов неограниченно росли. Можно использовать logrotate, чтобы заархивировать старую зарегистрированную информацию. Обычно logrotate выполняется как ежедневное задание cron. logrotate позволяет выполнять автоматическую ротацию, сжатие, удаление и отправку по почте файлов системного журнала. Каждый файл системного журнала может быть обработан ежедневно, еженедельно, ежемесячно или только тогда, когда становится слишком большим.

Поведением logrotate управляет файл конфигурации /etc/logrotate.conf (или специально определенный другой файл). Файл конфигурации может содержать и глобальные опции и опции, определенные для файла. Вообще, заархивированные сообщения сохраняются в течении конечного периода времени, и им присваиваются последовательные резервные названия. Например, одна из моих систем содержит следующие файлы в результате ее списка ротации:

```
-rw-r-----  1 root adm 4135 2005-08-10 04:00 /var/log/syslog
   -rw-r-----  1 root adm 6022 2005-08-09 07:36 /var/log/syslog.0
   -rw-r-----  1 root adm  883 2005-08-08 07:35 /var/log/syslog.1.gz
   -rw-r-----  1 root adm  931 2005-08-07 07:35 /var/log/syslog.2.gz
   -rw-r-----  1 root adm  888 2005-08-06 07:35 /var/log/syslog.3.gz
   -rw-r-----  1 root adm 9494 2005-08-05 07:35 /var/log/syslog.4.gz
   -rw-r-----  1 root adm 8931 2005-08-04 07:35 /var/log/syslog.5.gz
```

Упаковывание программного обеспечения

В начале был tarball

Для настройки программного обеспечения в Linux требуется гораздо меньше усилий, чем вы могли бы подумать. В Linux есть довольно четкий стандарт того, где файлы различных типов должны находиться, и установка специального программного обеспечения, в сущности, требует простого размещения правильных файлов в правильных местах.

Инструмент Linux tar (для "ленточных архивов", даже если не нужно, и обычно не нужно, использует формат ленты) прекрасно подходит для создания архивов

файлов с указанными местоположениями в файловой системе. Для дистрибутива вы можете сжать архив tar при помощи gzip (или bzip2). См. заключительную главу этого руководства о резервном копировании для получения дополнительной информации об этих утилитах. Сжатый tar архив обычно обозначают расширениями .tar.gz или .tgz (или .tar.bz2).

Ранние дистрибутивы Linux (и некоторые современные, такие как Slackware) используют простой tar архив как механизм развертывания дистрибутива. Для специализированных дистрибутивов собственного программного обеспечения систем Linux, поддерживаемых централизованно, такой способ остается самым доступным.

Специальные форматы архивов

Различные языки программирования и другие инструменты, появляющиеся вместе со специальными дистрибутивами, обычно не имеют предпочтений между дистрибутивами Linux и даже другими операционными системами. Python имеет свой собственный формат архива и инструменты distutils; Perl использует архивы CPAN; Java использует .jar файлы; Ruby использует .gem. Большинство неязыковых приложений имеют стандартную систему для установки плагинов или других инструментов для расширения базовых возможностей приложений.

Например, чтобы установить пакет Python вы можете использовать такой формат пакета как DEB или RPM, но часто разумнее следовать тому стандарту, для которого пакет создан. Конечно, для системных утилит и приложений и для большинства пользовательских приложений, тот стандарт, который используется, и есть стандарт дистрибутива Linux. Но для некоторых инструментов, написанных на специальных языках программирования, проще использовать какой-то ваш специальный инструмент вместо предложенного дистрибутивом или платформой (или внутреннее, или внешнее использование является конечной целью).

"Большая двойка" форматов пакетов

Есть два главных формата пакетов, используемые дистрибутивами Linux: Redhat Package Manager (RPM) и Debian (DEB). Оба они подобны в своих целях, но различны в деталях. Это форматы для "расширенного" архива файлов. Продвинутость, обеспеченная этими форматами пакетов, включает аннотации для номера версии, зависимости одного приложения от других приложений или библиотек, удобочитаемые описания инструментов пакета и общий механизм для управления установкой, обновлением или удалением инструментов пакета.

В формате DEB, вложенный конфигурационный файл control содержит большинство метаданных пакета. Для файлов RPM, эту роль играет файл spec. Более полные детали правильного создания пакетов в любом формате находятся вне этого учебного пособия, а здесь мы выделим только основы.

Что находится в .deb файле?

Пакет DEB создан при помощи инструмента архивирования, родственника tar -- ar (или другим более высокоуровневым инструментом, который использует ar). Поэтому мы можем использовать только ar, чтобы увидеть, что находится в .deb файле. Обычно используют высокоуровневые инструменты, такие как dpkg, dpkg-deb

или apt-get, чтобы фактически работать с пакетом DEB. Например:

```
% ar tv unzip_5.51-2ubuntu1.1_i386.deb
   rw-r--r-- 0/0      4 Aug  1 07:23 2005 debian-binary
   rw-r--r-- 0/0   1007 Aug  1 07:23 2005 control.tar.gz
   rw-r--r-- 0/0 133475 Aug  1 07:23 2005 data.tar.gz
```

Debian-бинарный файл просто содержит версию DEB (в настоящее время 2.0). Архив data.tar.gz содержит собственно файлы приложений - исполняемые, документацию, страницы помощи, файлы конфигурации и так далее.

Архив control.tar.gz представляется самым интересным. Давайте посмотрим на пакет DEB, который мы выбрали:

```
% tar tvfz control.tar.gz
   drwxr-xr-x root/root         0 2005-08-01 07:23:43 ./
   -rw-r--r-- root/root       970 2005-08-01 07:23:43 ./md5sums
   -rw-r--r-- root/root       593 2005-08-01 07:23:43 ./control
```

Как мы могли бы ожидать, md5sums содержит смесь шифров всех файлов дистрибутива в целях проверки. Файл control сообщает, где находятся метаданные. В некоторых случаях вы могли бы также захотеть включить в control.tar.gz скрипты postinst и prerm, чтобы принять специальные меры после установки или, соответственно, перед удалением.

Создание control-файла DEB

Инсталляционные скрипты могут делать то, что мог бы делать скрипт shell. (Посмотрите на некоторые примеры в существующих пакетах, чтобы понять, что имеется в виду.) Но такие скрипты являются дополнительными, и часто в них нет необходимости. Для .deb пакета необходим его control-файл. Формат этого файла содержит различные поля метаданных, и лучше всего это можно проиллюстрировать на примере:

```
% cat control
   Package: unzip
   Version: 5.51-2ubuntu1.1
   Section: utils
   Priority: optional
   Architecture: i386

   Depends: libc6 (>= 2.3.2.ds1-4)
   Suggests: zip
   Conflicts: unzip-crypt (<< 5.41)
   Replaces: unzip-crypt (<< 5.41)
   Installed-Size: 308
   Maintainer: Santiago Vila <sanvila@debian.org>
   Description: De-archiver for .zip files
   InfoZIP's unzip program. With the exception of multi-volume
archives
    (ie, .ZIP files that are split across several disks using PKZIP's
/& option),
    this can handle any file produced either by PKZIP, or the
corresponding
    InfoZIP zip program.
```

```
 .
 This version supports encryption.
```

В основном, кроме специальных случаев, ваш control-файл должен выглядеть точно также, как и этот. Для неконкретизированных типов процессоров – либо скрипты, либо пакеты документации, либо исходный код - используют `Architecture: all`.

Создание DEB пакета

Создание пакета DEB происходит при помощи dpkg-deb. Мы не можем здесь раскрыть все секреты изготовления хороших пакетов, но основная идея в том, что необходимо создать рабочий каталог ./debian/, и поместить в него необходимое содержание перед запуском dpkg-deb. Вы можете пожелать установить соответствующие права на ваши файлы после установки. Например:

```
% mkdir -p ./debian/usr/bin/
   % cp foo-util ./debian/usr/bin             # copy executable/script
   % mkdir -p ./debian/usr/share/man/man1
   % cp foo-util.1 ./debian/usr/share/man/man1   # copy the manpage
   % gzip --best ./debian/usr/share/man/man1/foo-util.1
   % find ./debian -type d | xarg chmod 755      # set dir permissions
   % mkdir -p ./debian/DEBIAN
   % cp control ./debian/DEBIAN   # first create a matching 'control'
   % dpkg-deb --build debian      # create the archive
   % mv debian.deb foo-util_1.3-1all.deb #rename to final package name
```

Еще о создании DEB пакета

В предыдущем примере можно увидеть, что локальная структура каталогов внутри ./debian/ организовывается так, чтобы соответствовать намеченной инсталляционной структуре. Для создания хорошего пакета необходимо еще несколько пунктов.

- Вообще, вы должны создать файл, названный с частью названия вашего дистрибутива ./debian/usr/share/doc/foo-util/copyright (добавьте к названию пакета).
- Хорошее дело - создавать файлы ./debian/usr/share/doc/foo-util/changelog.gz и ./debian/usr/share/doc/foo-utils/changelog.Debian.gz.
- Инструмент lintian позволяет убедиться, что при создании пакета не было допущено ни одной из стандартных ошибок, он проверит пакет DEB на предмет сомнительных свойств. Не все, на что lintian жалуется, необходимо обязательно исправить; но если вы собираетесь расширять дистрибутив, будет неплохо, если вы уберете все проблемы, которые возникли.
- Инструмент fakeroot позволит вам создавать пакеты не как пользователь root и полезен для того, чтобы собирать пакеты с правильным владельцем. Обычно нужны инструменты, установленные как root, а не как индивидуальный пользователь, который, может быть, собирал пакет (lintian предупредит об этом). Вы можете достигнуть этого с:

  ```
  % fakeroot dpkg-deb --build debian
  ```

Что находится в .rpm файле?

RPM придерживается немного другой стратегии при создании пакетов, чем DEB. Его файл конфигурации называют spec, а не control, но файл spec делает больше, чем файл control. Все детали шагов, необходимых для предустановки, постустановки, предудаления и непосредственно установки, содержатся, как вложенные скрипты, в конфигурации spec. Фактически, формат spec даже содержит макросы для общих действий.

Пакеты RPM создают при помощи утилиты rpm -b. Например:

```
% rpm -ba foo-util-1.3.spec  # perform all build steps
```

Этот процесс сборки пакетов базируется не на специфически названных каталогах, как в случае с DEB, а скорее на каталогах, прописанных в более сложном файле spec.

Создание метаданных RPM

Основные метаданные в RPM очень похожи на аналогичные в DEB. Например, foo-util-1.3.spec мог бы содержать что-то похожее на следующее:

```
# spec file for foo-util 1.3
   Summary: A utility that fully foos
   Name: foo-util
   Version: 1.3
   Release: 1
   Copyright: GPL
   Group: Application/Misc
   Source: ftp://example.com/foo-util.tgz
   URL: http://example.com/about-foo.html
   Distribution: MyLinux
   Vendor: Acme Systems
   Packager: John Doe <jdoe@acme.example.com>

   %description
   The foo-util program is an advanced fooer that combines the
   capabilities of OneTwo's foo-tool and those in the GPL bar-util.
```

Скрипты в RPM

Несколько разделов в RPM spec-файле могут содержать небольшие скрипты shell. Они включают:

- `%prep`: Шаги, которые нужно предпринять, чтобы получить готовую сборку, например, удалить более ранние сборки. Часто следующий макрос полезен и достаточен:

```
%prep
%setup
```

– `%build`: Шаги, чтобы фактически собрать пакет. Если вы используете средство make, то можно написать:

```
%build
 make
```

- `%install`: Шаги, чтобы установить пакет. И опять, если вы используете make, это могло бы означать:

```
%install
 make install
```

- `%files`: Вы должны включать список файлов, которые являются частью пакета. Даже если бы ваш Makefile использовал эти файлы, то менеджер пакетов (rpm) не будет знать о них, если вы не включите их сюда:

```
%files
 %doc README
 /usr/bin/foo-util
 /usr/share/man/man1/foo-util.1
```

Операции резервного копирования

О резервном копировании

Первое правило при создании резервных копий: Сделайте это! Слишком легко при администрировании сервера или настольной машины Linux пренебречь резервным копированием при составлении списка ваших потребностей.

Самый простой способ наладить систематическое резервное копирование – это настроить его, как одну из задач `cron`. См. Тему 213 нашего учебного пособия, где обсуждается конфигурирование crontab. Некоторым образом расписание резервного копирования зависит от инструментов, с помощью которых вы собираетесь его производить, и носителей, которые вы собираетесь использовать.

Резервное копирование на ленту – традиционный способ, и ленточные устройства продолжают предлагать наибольшую вместимость относительно недорогих носителей. Но в последнее время повсеместно стали использоваться записываемые и перезаписываемые компакт-диски и DVD-диски, и часто становится разумным использовать такие сменные носители для резервных копий.

Что подлежит резервному копированию

Что в системе Linux хорошо, так это то, что в ней используется определенное, иерархическое построение файлов. Как следствие, вам нет необходимости часто

делать копию всей иерархии файловой системы; большая часть ее может быть заново установлена с вашего дистрибутива Linux достаточно легко. В больших структурах образ мастер-сервера мог бы быть взят за основу системы Linux, которая, в свою очередь, уже могла бы быть настроена при помощи восстановления нескольких специально отобранных файлов, которые и подлежали бы резервному копированию.

В основном, то, что вы хотите сохранить – это каталоги /home/, /etc/, /usr/local/ и, возможно, /root/ и /boot/. Часто хотят также сделать копию некоторой части /var/, а именно /var/log/ и /var/mail/.

Резервное копирование при помощи cp и scp

Может быть, самый простой способ сделать резервную копию – использовать cp или scp с опцией -r (рекурсивно). Первая из этих команд копирует на локальные ресурсы (но включая монтирования NFS), а вторая может копировать на удаленные серверы надежно зашифрованным способом. В любом случае, вы должны иметь смонтированный ресурс с достаточным количеством свободного места, чтобы разместить файлы, которые вы хотите скопировать. Чтобы быть действительно застрахованным, резервная копия ваших данных должна быть размещена на другом физическом ресурсе.

Копирование при помощи cp или scp может быть только одной из частей большого списка резервного копирования. Здесь можно схитрить с помощью утилиты find, чтобы выяснять, какие файлы были изменены недавно. В следующем простом примере мы копируем из /home/ все файлы, которые были изменены за день:

```
#!/bin/bash

   # File: backup-daily.sh
   # ++ Run this on a daily cron job ++
   #-- first make sure the target directories exist
   for d in `find /home -type d` ; do mkdir -p /mnt/backup$d ; done
   #-- then copy all the recently modified files (one day)
   for f in `find /home -mtime -1` ; do cp $f /mnt/backup$f  ; done
```

Команда `cp -u` немного похожа, но более зависима от целостности файловой системы, куда происходит копирование между backup'ами. Существует рецепт, при котором эта команда прекрасно работает: замените точку монтирования /mnt/backup на другой адрес NFS. Найденный способ также хорошо работает и с `scp`, если вы зададите для удаленного ресурса информацию, необходимую для входа в систему.

Резервное копирование при помощи tar

Хотя cp и scp подходят для создания резервной копии, инструмент tar имеет более широкое использование, так как разработан специально для создания ленточных архивов. Несмотря на название, tar одинаково подходит для создания, как простого .tar файла, так и для записи необработанных данных на ленточное устройство. Например, вы могли бы создать копию на ленточном устройстве, используя команду:

```
% tar -cvf /dev/rmt0 /home    # Archive /home to tape
```

Внесением небольших изменений направляем выход в файл:

```
% tar -cvf /mnt/backup/2005-08-12.tar /home
```

Фактически, так как инструмент gzip может быть включен в поток, вы можете легко сжать архив в процессе создания:

```
% tar -cv /home | gzip - > /mnt/backup/2005-08-12.tgz
```

Вы можете комбинировать tar таким же образом, как было показано для cp или scp. Чтобы составить список файлов на ленточном устройстве, вы могли бы использовать:

```
% tar -tvf /dev/rmt0
```

Чтобы восстановить определенный файл:

```
% tar -xvf /dev/rmt0 file.name
```

Резервное копирование при помощи cpio

Утилита cpio - это мощная производная от tar. cpio работает с архивами tar, но также и с другими форматами и, кроме того, имеет множество встроенных опций. Cpio можно использовать с большим количеством аргументов, позволяющих отфильтровать скопированные файлы, и даже имеет встроенные аргументы, поддерживающие удаленное резервное копирование (вместо того, чтобы использовать канал с применением scp и др.) Главное преимущество, которое имеет cpio по сравнению с tar, что вы можете как добавить файлы к уже имеющемуся архиву, так и удалить файлы из архива.

Вот несколько примеров использования cpio:

- Создаем файловый архив на ленточном устройстве: % find /home -print |cpio -ocBv /dev/rmt0.
- Составляем список записей в файловом архиве на ленточном устройстве: % cpio -itcvB < /dev/rmt0.
- Восстанавливаем файл из ленточного устройства: % cpio -icvdBum file.name < /dev/rmt0.

Резервное копирование при помощи dump и restore

Чтобы сделать копию всей файловой системы сразу, иногда используется ряд инструментов, типа dump и restore (или их производных). К сожалению, эти инструменты являются специфическими для разных типов файловых систем и не всегда пригодны. Например, оригинальные dump и restore подходят только для ext2/3 файловых систем, в то время как инструменты xfsdump и xfsrestore используются для файловых систем XFS. Не каждый тип файловой системы имеет подходящую версию инструмента, но даже если они работают, аргументы могут

быть разными.

Полезно быть осведомленным об этих утилитах, но они не равнозначны для разных систем Linux. В некоторых случаях, например, если вы используете только разделы XFS, использование dump и restore может быть гораздо полезнее, чем использование простого tar или cpio.

Расширенное резервное копирование при помощи rsync

rsync - утилита, которая обеспечивает быструю расширенную передачу файлов. Часто для автоматизированного удаленного резервного копирования rsync является наилучшим инструментом для работы. Хорошая особенность rsync по сравнению с другими инструментами - то, что rsync может произвольно предписать двухстороннюю синхронизацию. Таким образом, вместо того, чтобы просто копировать наиболее новые или измененные файлы, rsync может автоматически удалить из отдаленной резервной копии файлы, удаленные на локальной машине.

Чтобы понять смысл аргументов, полезно посмотреть на этот не очень сложный скрипт (расположенный на Web-страницах rsync):

```
      #!/bin/sh
      # This script does personal backups to a rsync backup server. You
will
      # end up with a 7 day rotating incremental backup. The incrementals
will
      # go into subdirs named after the day of the week, and the current
      # full backup goes into a directory called "current"
      # tridge@linuxcare.com
      # directory to backup
      BDIR=/home/$USER
      # excludes file - this contains a wildcard pats of files to exclude
      EXCLUDES=$HOME/cron/excludes
      # the name of the backup machine
      BSERVER=owl
      # your password on the backup server
      export RSYNC_PASSWORD=XXXXXX
      BACKUPDIR=`date +%A`
      OPTS="--force --ignore-errors --delete-excluded --exclude-
from=$EXCLUDES
        --delete --backup --backup-dir=/$BACKUPDIR -a"
      export PATH=$PATH:/bin:/usr/bin:/usr/local/bin
      # the following line clears the last weeks incremental directory
      [ -d $HOME/emptydir ] || mkdir $HOME/emptydir
      rsync --delete -a $HOME/emptydir/ $BSERVER::$USER/$BACKUPDIR/
      rmdir $HOME/emptydir
      # now the actual transfer
      rsync $OPTS $BDIR $BSERVER::$USER/current
```