электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.451.5

Файловая система для просмотра информации и управления удалёнными компьютерами через SaltStack

***Инфлянскас Р.В.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»

Научный руководитель: Волкова Л.Л., ассистент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Программное обеспечение ЭВМ и информационные технологии»
irudakov@bmstu.ru

Один из базовых принципов Unix гласит: «Всё является файлом». Linux, унаследовавший этот и многие другие принципы Unix, различает много типов файлов в дополнение к стандартным файлам и каталогам.

В ядре Linux, помимо дисковых файловых систем (наиболее используемыми из которых являются `btrfs`, `ext3`, `ext4`, `ReiserFS`, `XFS`), присутствуют также системы, предоставляющие доступ к ресурсам компьютера, отличным от данных на жёстком диске. `procfs` позволяет получить доступ к информации о системных процессах из ядра (она необходима для выполнения таких команд, как `ps`, `w`, `top`), `sysfs` экспортирует в пространство пользователя информацию ядра Linux о присутствующих в системе устройствах и драйверах, `ramfs` используется для создания RAM-диска в процессе загрузки системы, `tmpfs` поддерживает работу с виртуальной памятью. Для Linux также существуют сетевые системы: `SMB` (реализующая поддержку “общих папок Windows”), `NFS` (популярный в Unix-подобных системах протокол сетевого доступа к файловым системам), а также кластерные системы `Lustre`, `Cerp`, `GlusterFS` и другие. Реализованы также системы для обеспечения абстракции над сетевыми ресурсами: `sshfs` (доступ к файлам по протоколу `SSH`), `ftpfs` (доступ к файлам по протоколу `FTP`).

Не существует широко известных файловых систем для просмотра информации и удалённого управления компьютерами. В связи с этим было решено расширить принцип «всё есть файл», создав файловую систему, в которой файлами являлись бы параметры удалённых компьютеров и действия над ними.

Метод реализации файловой системы

Существует три основных способа реализации файловой системы Linux: изменение исходного кода ядра, создание загружаемого модуля ядра, использование FUSE [1].

Изменение исходного кода ядра

Изменение исходного кода ядра является наиболее затратным по временным ресурсам, поскольку при каждом изменении файловой системы необходимо пересобирать ядро. Также этот вариант не позволяет выпускать новые версии файловой системы чаще, чем новые версии ядра. К тому же файловая система для удалённого управления скорее всего не нужна рядовому пользователю и увеличит и так “раздутое и огромное” (Линус Торвальдс, 2009 г.) ядро.

Создание загружаемого модуля ядра

Файловая система оформляется в виде загружаемого модуля ядра, компилируемого в бинарный файл формата kernel object (.ko), который загружается и выгружается командами `insmod` и `rmmod`.

Этот подход позволяет вести разработку быстрее (сборка модуля ядра занимает на порядки меньше времени, чем сборка всего ядра) и не зависеть от выпусков ядра. Недостатком являются ограниченные возможности по доступу к объектам ядра [2].

Использование FUSE

FUSE — загружаемый модуль для Unix-подобных ОС, который позволяет пользователям без привилегий создавать их собственные файловые системы без необходимости переписывать код ядра. Это достигается за счёт запуска кода файловой системы в пространстве пользователя, в то время как модуль FUSE только предоставляет мост для актуальных интерфейсов ядра. FUSE была официально включена в главное дерево кода Linux в версии 2.6.14.

К недостаткам этого подхода относится то, что предоставляя унифицированный интерфейс к созданию файловых систем многих ОС, FUSE не предоставляет доступа к низкоуровневым объектам реализации слоя файловой системы (VFS в Linux), что сказывается на производительности.

Выбор метода реализации файловой системы

Управление большим количеством удалённых компьютеров должно производиться максимально быстро. Поэтому было решено отказаться от использования FUSE. Так как доступ к объектам ядра, возможный только из самого ядра, не требовался, было решено реализовать файловую систему в виде модуля ядра.

Метод удалённого управления

Программирование в пространстве ядра накладывает существенные ограничения. Использование стандартной библиотеки C и библиотек GNU не представляется возможным, поэтому использование сокетов Беркли для сетевого взаимодействия является трудновыполнимой задачей. В связи с этим было принято решение осуществлять сетевое взаимодействие в пространстве пользователя и лишь затем передавать данные в пространство ядра для обработки модулем файловой системы.

Существует большое количество ПО удалённого управления. В связи с этим представляется более разумным не создавать ещё одно ПО для удалённого управления, а приспособить уже имеющееся. Категория ПО, ориентированная на массовое управление серверами, отображение и изменение их конфигурации, называется “ПО для конфигурационного управления” (англ. configuration management software). Именно такое ПО отвечает требованиям, поставленным перед файловой системой.

Подробное сравнение свободного ПО для конфигурационного управления проведено в [3]. В качестве ПО для конфигурационного управления был выбран SaltStack (сокращённо — Salt), вследствие того что он обладает свободной лицензией (Apache), поддерживает большое количество ОС (включая GNU/Linux, OS X и Microsoft Windows), а также имеет высокую скорость работы, обусловленную использованием сетевой библиотеки ZeroMQ и протокола обмена сообщениями MessagePack, ориентированных на производительность. В связи с этим файловой системе было дано название `saltfs`.

Использование Salt

В системе управления Salt используются следующие понятия:

Зерно (англ. grain) — статическая (меняющаяся не чаще, чем несколько раз во время работы ОС) информация о компьютере, ОС и конфигурации.

Функция — действие, которое возможно осуществить над клиентским компьютером. Примеры: test.ping (проверка наличия миньона в сети), system.shutdown

(выключение ОС), pkg.install NAME (установка пакета из репозитория).

Мастер (англ. master) — серверный компьютер, на котором запущена программа salt-master. Обычно в системе Salt находится один мастер.

Миньон (англ. minion) — клиентский компьютер, на котором запущена программа salt-minion, управляемая мастером. В системе Salt может находиться произвольное количество миньонов (в наиболее крупных системах Salt находятся десятки тысяч миньонов).

Модуль — группа функций, схожих по смыслу. Примеры: `test` (модуль с функциями проверки работы системы Salt), `system` (модуль управления питания компьютера и основными параметрами ОС), `pkg` (модуль для управления установленным ПО).

Иерархия файловой системы

На рисунке 1 изображена иерархия файловой системы.

Первый уровень

На верхнем уровне находятся директории, именованные идентификаторами миньонов.

Второй уровень

На втором уровне находятся директории, соответствующие модулям. В общем случае они различны для разных платформ: зависят от установленной ОС (например, модуль `linux_acl` для GNU/Linux) и прикладного ПО (модули `postgres` и `mysql` присутствуют при установленных PostgreSQL и MySQL).

Третий уровень

На третьем уровне находятся файлы, соответствующие функциям. Для каждой функции определены операции чтения и записи. При чтении происходит выдача документации по заданной функции. Запись параметров функции служит для вызова функции.

Для управления гранулами в Salt служит модуль `grains`. Он обрабатывается особым образом. В нём, в отличие от определённых в Salt функций, содержатся файлы, соответствующие гранулам.

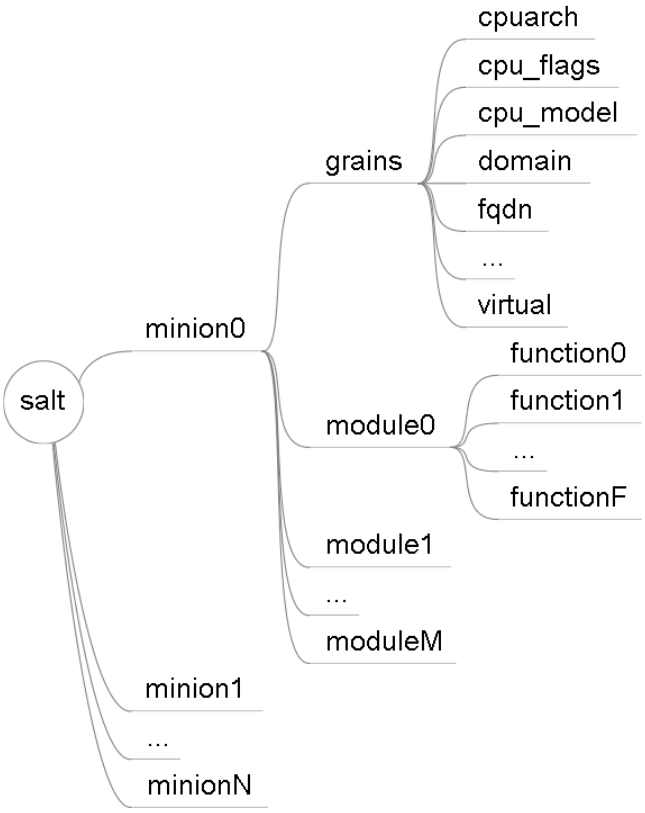

Рис.1. Иерархия файловой системы

Устройство файловой системы

Файловая система `saltfs` не связана с физическими устройствами хранения файлов, и все файлы в ней создаются “на лету”. Для получения списка директорий, содержимого файлов, вызова функций используется скрипт командной оболочки `fish`, который вызывает программу `salt`. На рисунке 2 показано взаимодействие файловой системы с пользовательским ПО.

Существует несколько способов организации взаимодействия модуля ядра с пространством пользователя [4]: `procfs`, `sysfs`, `configfs`, `debugfs`, `sysctl`, символьные устройства, `netlink`-сокеты, системные вызовы ядра, `mmap`.

В качестве механизма взаимодействия модуля ядра с пространством пользователя был выбран обмен данными с помощью `procfs` из-за простоты его использования, отсутствии необходимости написания дополнительного пользовательского приложения.

В качестве способа запуска скрипта, вызывающего программу `salt`, было использовано API `usermode_helper` [5].

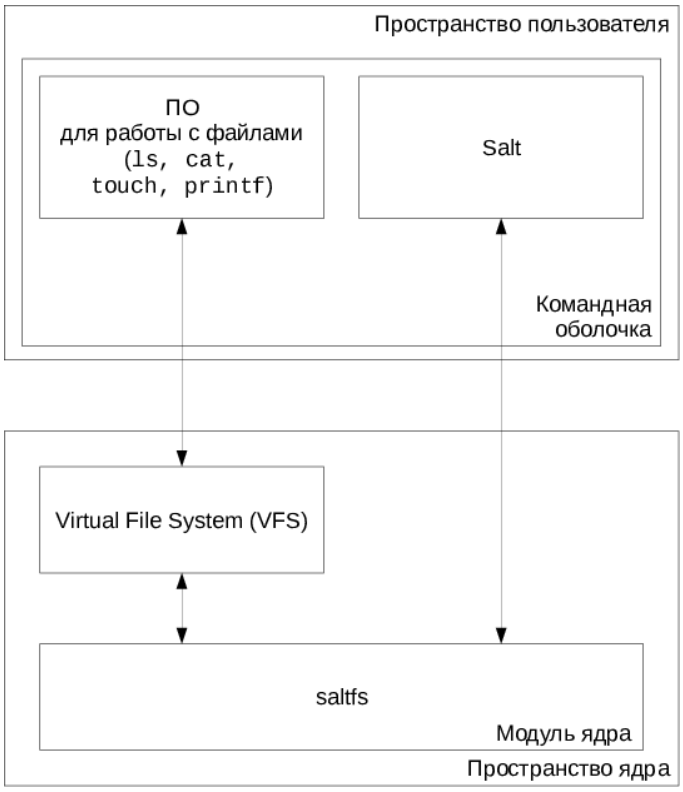

Рис. 2. Взаимодействие файловой системы с ПО

Пример использования

Для разработки и тестирования файловой системы использовалась среда виртуализации VirtualBox под управлением ПО для создания и конфигурирования виртуальной среды разработки Vagrant. Был создан стенд, состоящий из двух виртуальных машин: openSUSE 13.2 и Microsoft Windows 7.

Пример интерактивной сессии:

```
Монтирование файловой системы
# mount -t saltfs saltfs /salt
Получение списка минионов (директории в папке salt)
# echo /salt/*/
vbox-opensuse vbox-win7
# cd /salt/
Установка vim на минион vbox-opensuse
# echo vim > vbox-opensuse/pkg/install
Результат установки
# cat result
vbox-opensuse:
    ----------
    vim:
        ----------
        new:
            2:7.3.547-7
```

```
        old:
            1
```
Перезагрузка миниона ***vbox-****
```
# touch vbox-*/system/restart
```
Получение списка ***ip*** *адресов миниона* ***vbox-opensuse***
```
# cat vbox-opensuse/grains/ipv4
vbox-opensuse:
    - 127.0.0.1
    - 10.0.2.15
    - 10.10.0.2
```
Получение справки
```
# cat vbox-opensuse/network/ping
network.ping:

    Performs a ping to a host

    CLI Example:

        salt '*' network.ping archlinux.org
```

Вывод

Была разработана файловая система для просмотра информации и удалённого управления компьютерами. Файловая система позволяет системным администраторам пользоваться принципом «в Unix всё есть файл» не только для управления локальным, но и удалёнными компьютерами. Использование системы конфигурационного управления SaltStack, лежащей в основе файловой системы `saltfs`, позволило получить сотни функций для управления компьютерами, начиная от проверки наличия компьютера в сети (`test.ping`), заканчивая управлением `docker`, `handoop` и `openstack`, под единым интерфейсом.

Список литературы

1. Love R. Linux kernel development. 3rd ed. Boston: Addison-Wesley, 2010. 467 p.
2. Salzman P.J., Burian M., Pomerantz O. The Linux Kernel Module Programming Guide. CreateSpace, 2009. 86 p.
3. Авторы Wikipedia. Comparison of open-source configuration management software. Режим доступа: https://en.wikipedia.org/wiki/Comparison_of_opensource_configuration_management_software (дата обращения: 01.03.2015).
4. Keller A. Kernel Space - User Space Interfaces. 2008. Режим доступа:

http://people.ee.ethz.ch/~arkeller/linux/kernel_user_space_howto.html (дата обращения: 01.03.2015).

5. Jones M. T. Invoking user-space applications from the kernel // Kernel APIs. № 1. 2010. Режим доступа: http://www.ibm.com/developerworks/library/l-user-space-apps (дата обращения: 01.03.2015).
6. Embedded Linux Experts. Linux Cross Reference. 2014. Available at: http://lxr.free-electrons.com/, accessed 01.03.2015.