

# Простой скрипт для резервного копирования

**Рашид Ачилов**

Он умеет скопировать каталог с жесткого диска в заранее обусловленное место, предварительно его запаковав. Он может создать резервную копию части сетевого ресурса Windows, для чего он сам подключит и сам отключит ресурс. Он способен зарезервировать данные из СУБД MySQL (правда, только формата MyISAM). Он не перепутает каталоги для хранения копий и никогда не затрет старую копию более новой. Он вежлив – по завершении работы непременно доложит о выполнении. Он – это скрипт для резервного копирования `130.backup-dirs`.

## Изобретаем велосипеды

История разработки скрипта восходит к началу моего осмысленного увлечения UNIX, как минимум к 1998 году. Начинаящие системные администраторы обычно познают ценность резервных копий после первого выхода из строя жесткого диска, и я не был исключением. И непосредственно после восстановления системы (слава Богу, вышла из строя рабочая машина, не сервер) возникла мысль «а не написать ли мне скриптик, который бы

по расписанию копировал некоторые каталоги в определенное место?». С тех пор скрипт многократно перерабатывался, однажды был коренным образом переписан с `tcsh` на `/bin/sh`, расширился, научился работать с сетью Windows и базами MySQL, но суть его всегда оставалась неизменной – взять каталог, запаковать его и переписать в заранее оговоренное место, выполнив перед этим и после этого некоторые действия, направленные на сохранение целостности (подключение-от-

ключение ресурсов, блокировка-разблокировка таблиц БД).

Эта статья может принести пользу как тем, кто ищет практические советы, так и людям любознательным, которым интересно, как оно устроено и работает.

В первой части объясняется, как установить и настроить скрипт и как его запустить. Во второй части разбираются некоторые фрагменты из исходного кода, которые мне показались заслуживающими внимания.

Здесь сразу отмечу, что скрипт работает не сам по себе, а достаточно глубоко интегрирован с другими разработками – [1, 2] (mountsmb2), хотя, конечно, ни одна из них не является критичной для работы всего скрипта. Но в отсутствии настроенного mountsmb2 не будет доступна возможность копировать данные с сетевых ресурсов Windows.

## Использование

Итак, что нужно сделать для того, чтобы скрипт работал. Сначала его, разумеется, нужно скачать. Загружается он с [3], из раздела «Скрипты». На самом деле похожих файлов там три, но они абсолютно одинаковы, за исключением имен переменных, используемых для настройки. Оттуда же загружаются:

- основной конфигурационный файл `rmbbackup.conf`;
- дополнительный конфигурационный файл `dbconnect.conf` (последний нужен только для резервирования баз данных MySQL).

Вообще-то никакого конфигурационного файла у скрипта нет, `rmbbackup.conf` следовало бы называть файлом описания точек резервирования – в нем перечисляются каталоги, которые нужно копировать, и указывается временной интервал. Впрочем, формат этого файла будет подробно описан ниже. А все основные переменные заносятся в файл `/etc/periodic.conf`. Это, конечно, не очень удобно и будет переделано в следующих версиях. Здесь надо еще отметить, что в отсутствие файла `rmbbackup.conf` скрипт не запускается. Для запуска скрипта нужно добавить в `/etc/periodic.conf` следующие строки:

```
daily_backup_fileplace="/path/to/backup"
daily_backup_dirs_cfg="/path/to/backup/dirs/config"
daily_backup_owner="owner"
daily_backup_group="group"
daily_backup_mode="octal mode like 0644"
daily_backup_dirmode="octal mode like 0755"
daily_backup_startup="/path/to/startup/scripts"
daily_backup_connect="username"
daily_backup_dbconnect_conf="/path/to/dbconnect.conf"
```

Обязательными являются все параметры, кроме двух последних.

- **daily\_backup\_fileplace** – задает корневой каталог, где будут размещаться все каталоги и подкаталоги с файлами резервных копий. При выборе значения `daily_backup_fileplace` следует помнить, что размещение файлов там делается простым копированием, поэтому здесь не может быть задан путь к съемному носителю.
- **daily\_backup\_dirs\_cfg** – задает размещение основного файла с описаниями точек резервирования `rmbbackup.conf`. Если этого файла не существует, скрипт не работает.
- **daily\_backup\_owner**, **daily\_backup\_group** – задают владельца и группу владельцев архивных файлов.
- **daily\_backup\_mode** и **daily\_backup\_dirmode** – соответственно указывают права, устанавливаемые на создаваемые файлы и каталоги.
- **daily\_backup\_startup** – фиксирует путь к каталогу, откуда будет запускаться скрипт останова программы, если таковой указан для резервирования катало-

га. Сделано это для того, чтобы перед началом копирования все файлы были закрыты. Скрипт берет путь `daily_backup_startup` и запускает в нем указанную программу с параметрами: перед началом копирования – `stop`, после окончания копирования – `start`.

- **daily\_backup\_connect** – задает пользователя, от имени которого будет выполняться подключение к сетевому ресурсу. Подключение выполняется с использованием `mount_smbfs`, со всеми его огромными недостатками, поэтому перем тем, как начать резервировать сетевые ресурсы, необходимо настроить `nsmb.conf` и прочие упоминаемые в `man mount_smbfs` файлы. Обязательно также наличие файла `.mssmbrc` в домашнем каталоге пользователя, от имени которого запускается скрипт, описывающего точки монтирования для подключения всех резервируемых ресурсов.
- **daily\_backup\_dbconnect\_conf** – задает путь к файлу с описанием настроек для подключения к СУБД MySQL для резервирования баз данных. На самом деле, конечно, копирование тут ни при чем, в СУБД отправляется команда `lock tables <таблицы заданной базы данных>`, потом делается копирование и тут же `unlock tables`. В отличие от всех других типов резервируемых данных при резервировании БД MySQL блокировка таблиц делается только на время, необходимое для того, чтобы скопировать файлы БД в рабочий каталог. Упаковка каталога будет делаться уже после разблокировки таблиц.

Внимательный читатель тут же может задать вопрос: «А почему бы для копирования баз MySQL не использовать `mysqldump`? Ведь его разработали именно для этого?». Разумеется, вариант с `mysqldump` приходит в голову первым. И мне он тоже пришел в голову. Но скрипт писался для резервирования базы данных биллинга, которая пополняется 24 часа 7 дней в неделю каждые пять минут, и нужно было иметь твердую гарантию, что прием данных будет блокироваться на время, пока работает скрипт. Для этого был проведен быстрый анализ исходного кода `mysqldump`, и, после того, как было обнаружено, что в нем не выполняется никакая блокировка, пришлось придумать вот такой способ.

Наличие файла `rmbbackup.conf` для работы скрипта обязательно. Это простой текстовый файл свободного формата. Он состоит из трех секций, имена которых заключены в квадратные скобки, а между ними по одному ресурсу на строку перечислены точки резервирования.

Например:

```
[daily]
/tmp/smbfs/office/1cbase|/Manfc/Mnfcctr:smbfs
test:mysql

[weekly]
/etc:ufs
/usr/local/etc:ufs
/usr/src/sys/i386/conf:ufs

[monthly]
/var/log/squid:ufs:squid.sh
```

Разумеется, все приведенные здесь пути – условные. Общий формат описания точки резервирования следующий:

```
<путь[|отделяемая часть]>:<тип файловой системы>: ↵
<скрипт перезапуска>
```

где <путь> – это полный путь к резервируемому ресурсу. Ресурс будет резервироваться путем упаковки в архив tar.bz2 с присвоением имени архиву, образованному путем замены в полном пути к ресурсу символа разделения путей (слэш прямой) на символы подчеркивания. Таким образом, например каталог /usr/local/etc будет упакован в файл \_usr\_local\_etc.tar.bz2.

**Внимание!** Использовать каталоги с именами, содержащими пробелы, в качестве точек резервирования нельзя. Наличие в имени символов подчеркивания не влияет на работу скрипта, разве что имена файлов получатся необычные.

Отделяемая часть имеет смысл только для сетевых ресурсов Windows. Изначально можно было резервировать только весь разделяемый ресурс, что было очень неудобно. Для того чтобы резервировать часть дерева каталогов, и был введен параметр «отделяемая часть». Он отделяет в пути к ресурсу точку монтирования, в которую будет подключаться ресурс от части полного пути к каталогу, который будет резервироваться. Таким образом, в рассмотренном выше примере /tmp/smbfs/office/1cbase является точкой монтирования, куда будет подключен ресурс из .mssmbrc с такой же точкой монтирования, но резервироваться будет только каталог /tmp/smbfs/office/1cbase/Manfc/Mnfctr (и соответственно превратится в файл \_tmp\_smbfs\_office\_1cbase\_Manfc\_Mnfctr.tar.bz2). Если же резервируется база данных, то указывается только ее имя, расположение каталога БД задается в файле dbconnect.conf.

<тип файловой системы> может быть ufs, msdos, smbfs и mysql.

Если указано ufs, никаких дополнительных действий не производится.

Если указано msdos, производится поиск описания запрошенной точки резервирования среди смонтированных файловых систем. В случае, если заданный путь не был найден, выполняется поиск в файле /etc/fstab и, если он там описан, то монтируется, иначе выдается сообщение об ошибке и точка резервирования пропускается.

Если указано smbfs, выполняются действия, аналогичные msdos, только поиск проводится в файле .mssmbrc пользователя, запустившего скрипт.

Если указано mysql, то выполняется подключение к СУБД, выдается команда lock tables <перечень таблиц заданной БД>, копирование БД в рабочий каталог, unlock tables, потом рабочий каталог упаковывается.

<скрипт перезапуска> задает имя исполняемого файла, который перед началом резервирования остановит программу, выполняющую запись в файлы, а после завершения запустит эту программу заново. Полный путь к программе перезапуска формируется как \$daily\_backup\_startup/<имя\_скрипта>.

Наличие файла dbconnect.conf обязательно только для резервирования баз данных. В настоящий момент поддерживаются только базы данных СУБД MySQL формата MyISAM, формат InnoDB не поддерживается. Их резервировать достаточно просто – каждая БД в формате MyISAM

представляет собой каталог, в котором находятся файлы данных и индексов, этот каталог копируется в другое место и упаковывается. Чтобы исключить возможность записи в момент копирования таблиц БД, собственно говоря для этого и выполняется подключение к СУБД, само копирование выполняется командой ср.

Но поскольку СУБД ничего не знает о том, что идет копирование файлов базы, необходимо подключиться и заблокировать все таблицы БД, которая резервируется. Это делается с помощью команды lock tables. Поскольку в MySQL 4.x отсутствует команда, позволяющая заблокировать все таблицы, то список таблиц предварительно получается командой «show tables». После завершения копирования выдается команда unlock tables.

Сам формат файла простой – текстовый файл, так же, как и rmbbackup.conf, разделенный на секции. Имя секции должно совпадать с именем резервируемой базы данных.

```
[testmysql]
dbtype=mysql
user=testuser
password=testpwd
host=localhost
port=3306
location=/var/db/mysql
```

Обязательными являются только первые три параметра, остальные принимают значения по умолчанию. Пользователь, от имени которого будет выполняться подключение, прописанный в данном файле, должен иметь права на выдачу команды «LOCK TABLES» для указанной базы данных.

Непосредственно в самом скрипте не предусмотрено никаких средств, обеспечивающих запуск по времени. Для запуска используются возможности periodic – как правило, я создаю отдельный crontab для пользователя root (не пересекающийся с /etc/crontab) командой «crontab -е» и заново в него такую строчку:

```
0 1 * * * /usr/sbin/ ↵
periodic /usr/local/etc/periodic/daily
```

Правда, тут необходимо отметить следующее: запускать скрипт копирования от пользователя root, с моей точки зрения, имеет смысл только в том случае, если не выполняется резервирование сетевых ресурсов Windows. Если же выполняется, то мне кажется, лучше создать отдельного пользователя (я его называю, как правило, rmbbackup), настроить для него .mssmbrc, создать необходимые каталоги для монтирования сетевых ресурсов и выполнять скрипт от его имени. Хотя обязательным, конечно, это не является, параметр daily\_backup\_connect позволяет задать имя любого пользователя, имеющего права для подключения к сетевому ресурсу.

## Реализация

А теперь – информация для тех, кто, возможно, захочет улучшить скрипт, адаптировать его под свои нужды, или просто любознательных. Для точности изложения мы будем рассматривать скрипт суточного копирования.

Скрипт на самом деле несложен, но содержит несколько интересных приемов, в частности процедуру разбора конфигурационного файла.



```

parse_sectioned_config()
{
    # Удалить из файла комментарии
    bla=`awk '{if ($1 == "#") next; }' < $filename`

    # Разбить по первой квадратной скобке [, так что части
    # будут выглядеть типа: "section] /path/to /path/to"
    # "section] /path/another /path/more"
    saveifs=$IFS
    IFS=[
    set $bla

    # Взять одну часть "section] /path/to /path/more"
    for blabla in $bla
    do
        IFS=]
        set $blabla

    # Разбить по второй скобке, так что части будут выглядеть:
    # "section" "/path/to /path/more". Это нужно для проверки
    # имени секции
        if [ $1 = $section ]; then
            line=$2
            IFS=$saveifs
            return
        fi
    done
}

```

В результате чего мы имеем на выходе в переменной `line` перечень точек резервирования, заданный в данной секции. Недостатком данной процедуры является невозможность отбрасывания комментариев в том случае, если между знаком комментария и текстом не стоит пробела.

Процедуру создания каталога для хранения файлов описывать особого смысла не имеет – просто проверка на наличие корневого каталога для хранения данных (из переменной `daily_backup_fileplace`), каталогов с именами, состоящими из года, года и месяца, года, месяца и дня в форматах ГГГГ, ММ-ГГГГ, ДД-ММ-ГГГГ соответственно и установка на создаваемые каталоги прав и владельцев из соответствующих переменных `/etc/periodic.conf`.

Собственно резервирование выполняется в одном операторе `case`, в котором проверяется задание переменной `daily_backup_enable` – если она не установлена в `YES`, ничего не выполняется, скрипт просто завершается сразу же. Если же переменная задана, проверяется задание еще двух переменных – `daily_backup_dirs_cfg` и `daily_backup_fileplace`, кроме того, проверяется наличие файла, указанного в переменной `daily_backup_dirs_cfg`.

Если все переменные указаны, то разбирается указанный конфигурационный файл, в котором ищется секция `daily` и проверяется перечень полученных точек резервирования. Если полученный список пуст, то резервировать нечего и скрипт завершает работу.

Если же точки резервирования указаны, то скрипт работает в цикле по указанным путям. Описание точки резервирования разбирается на три части – путь, тип файловой системы и скрипт запуска/останова. Третья часть может быть не указана, при этом никакой ошибки не фиксируется, просто ничего не будет остановлено и перезапущено. Если же опущен тип файловой системы, фиксируется ошибка и скрипт переходит к обработке следующей точки резервирования.

Если в пути указана отделяемая часть, то выделяется точка монтирования, если указан скрипт останова/запуска, то выполняется останов сервиса, формирующего дан-

ные по указанному пути. Как уже упоминалось, путь к файлу останова формируется из переменной конфигурационного файла и имени указанного скрипта:

```

if [ ! ${#bstopper} -eq 0 ]; then
    dstop="yes"
    $daily_backup_startup/$bstopper stop
fi

```

Далее идет `case` по типам поддерживаемых файловых систем и выполнение подготовительных действий.

Для типа файловой системы `ufs` не выполняется никаких действий.

Для типа файловой системы `msdos` сначала выполняется поиск заданной точки резервирования в файле `/etc/fsab`. Если заданный путь отсутствует в `/etc/fstab`, то выдается сообщение об ошибке, скрипт переходит к обработке следующей точки резервирования. Если путь найден, то он монтируется, если смонтировать не удалось, скрипт переходит к обработке следующей точки резервирования.

Для типа файловой системы `smbfs` (сетевой ресурс Windows) подготовка идет дольше и сложнее. Сначала обнаруживается домашний каталог пользователя, запустившего скрипт, если этот пользователь создан без домашнего каталога (а такое возможно, если резервировать только локальные точки), то невозможно будет резервировать точки с типом `smbfs`. Если домашний каталог найден, в нем ищется файл `.mssmbrc` (справку по файлу `.mssmbrc` можно получить, установив пакет `sysutils/mountsmb2` и получив файл `README.FreeBSD`), если файл не найден, точка резервирования пропускается. Файл `.mssmbrc` нужен для того, чтобы найти сервер, с которого будет монтироваться данный ресурс. Когда ресурс найден, берется имя сервера, переводится в верхний регистр и ищется в файле `/etc/nsmb.conf` (в нем содержится указание на таблицы перекодировки, это важно в том случае, если резервируются файлы с русскими именами).

```

smbmp=`grep $bmount $homedir/.mssmbrc`
# Точка монтирования не найдена в файле .mssmbrc
if [ -z "$smbmp" ]; then
    echo "$bmount missed in $homedir/.mssmbrc, ␣
    mount aborted"
    rc=3
else
    # Разобрать запись из файла .mssmbrc
    set $smbmp
    # Перевести имя сервера в верхний регистр
    userver=`echo $1 | tr "[:lower:]" "[:upper:]"`
    gserver=`grep $userver /etc/nsmb.conf`
    if [ -z "$gserver" ]; then
        echo "$1 did not described in /etc/nsmb.conf, ␣
        mount aborted"
        rc=3
    fi

```

Если сервер найден, то нужно смонтировать заданную точку через `mount_smbfs`, причем в качестве пользователя, от чьего имени будет выполняться монтирование, используется параметр `daily_backup_connect`:

```

mounts=`mount | grep $bmount`
# Если точка резервирования не смонтирована
if [ -z "$mounts" ]; then
    # Это все переменные из записи файла .mssmbrc
    mount smbfs -f $5 -d $4 ␣
    //${daily_backup_connect}@$1/$2 $bmount
    if [ $? -ne 0 ]; then

```

```
status=$?
echo "$bmount cannot be mounted, error code is $status"
rc=3
else
domount="yes"
fi
fi
```

Для типа файловой системы `mysql` (хотя в этом случае правильнее говорить о типе БД, конечно) подготовка идет дольше всех. Первым шагом проверяется задание переменной `daily_backup_dbconnect_conf`. Если она не задана, точка резервирования пропускается. Потом ищется программа MySQL, с помощью которой команды будут передаваться в СУБД. Если она не найдена, точка резервирования пропускается. Если программа MySQL найдена, то с помощью процедуры `parse_sectioned_config` в файле, заданном в `daily_backup_dbconnect_conf` ищется секция с именем резервируемой базы и с помощью команды `eval` создаются переменные, описанные в файле. После этого скрипт запрашивает у СУБД список таблиц в резервируемой базе.

```
section=$bpath
filename=$daily_backup_dbconnect_conf
parse_sectioned_config
tokens=$line
# Выполнить распознавание всех переменных
for onetoken in $tokens
do
eval $onetoken
done
# Получаем список таблиц в базе данных, пропускаем первую
# строку ответа
tables=`$mysql -B -u $user -h $hostname -P $port \
-r$password -e "use $bpath; show tables;" | tail -n +2`
# Если попытка неудачна, возможно, неверно указан
# пользователь или пароль
status=$?
if [ $status -ne 0 ]; then
echo "Cannot read tables list from database $bpath, " \
"probably invalid authentication credentials"
rc=3
continue
fi
```

Для того чтобы заблокировать таблицы, нам необходимо сформировать их список в одну строку, точнее говоря, сформировать команду «`lock tables tablename1, tablename2,... tablenameN read`». Для этого достаточно взять переменную `tables` и сформировать одну длинную строку. Да только вот незадача – если формировать ее просто перечислением, то или в начале строки окажется лишняя запятая (`lock ,tablename read`), или лишняя запятая окажется в конце (`lock tablename, read`). В любом случае СУБД среагирует одинаково – ошибка. С помощью небольшой хитрости мы обходим эту ситуацию:

```
comma=0
locks=""
for onetable in $tables
do
# Если это не первый элемент в строке, приписать запятую
# и пробел
if [ ! $comma -eq 0 ]; then
locks="$locks, "
fi
locks="$locks $onetable read"
comma=1
done
```

И наконец самый важный участок кода в резервировании БД – мы формируем строку команды, которую переда-

ем в MySQL. Самым интересным в ней является то, что мы блокируем таблицы, выполняем системную команду копирования и снимаем блокировку таблиц в одной команде. Выглядит это немножко... нетривиально, но это единственный способ, который я нашел, чтобы таблицы оставались заблокированными на время копирования – как только кончается сеанс MySQL, так все блокировки автоматически снимаются.

```
printf "use $bpath;\nlock tables $locks;\n \\\! cp -Rp \
$location/$bpath $filepath;\nunlock tables;" | \
mysql -B -n -u $user -P $port -h $hostname \
-r$password
status=$?
if [ $status -ne 0 ]; then
echo "Cannot copy tables for base $bpath, " \
"copying routine returned $status"
rc=3
continue
fi
```

Обратите внимание на количество экранирующих символов в строке команды – поскольку команда дважды передается через шеллы, число экранирующих слэшей должно быть таким, чтобы символ «`!`», обозначающий в MySQL выполнение команды оболочки, распознавался соответствующим образом.


Вот теперь только можно начинать резервирование. Делается оно одним фрагментом кода для всех типов файловых систем (не зря ведь велась такая долгая подготовка):

```
tradir=`echo $bmount$bsepart | sed 's@/@_@g'`
tar -cyf "$filepath/$tradir.tar.bz2" $bmount$bsepart > \
/dev/null 2> /dev/null \
&& chown $daily_backup_owner:$daily_backup_group \
$filepath/$tradir.tar.bz2 \
&& chmod $daily_backup_mode \
$filepath/$tradir.tar.bz2 && rc=0 || rc=3
```

Первая строка – это замена в имени файла всех символов «`/`» на символ подчеркивания. Дальше уже все неинтересно – если точка резервирования монтировалась непосредственно скриптом, то скрипт же ее и размонтирует, если необходимо очистить рабочие файлы, они удаляются, если выполнялась остановка некоторого сервиса, он запускается заново. И собственно все.

## Заключение

Разумеется, этот скрипт далек от совершенства. Хотя бы потому, что он каждый раз создает новый каталог – для точного резервирования получается немыслимая избыточность. Он не умеет напрямую сбрасывать архивы на съемные носители, не умеет подключаться с разными именами. Но тем не менее я его использую уже более семи лет и еще ни разу не пожалел о том, что однажды написал его. Надеюсь, он пригодится и вам.

Удачи! 

1. Ачилов Р. Копирование файлов в автоматическом режиме с множества компьютеров через SSH. //Системный администратор, № 12, 2004 г. – С. 12-17.
2. Ачилов Р. FreeBSD в домене Windows: дополнительные возможности. //Системный администратор, № 1, 2007 г. – С. 62-69.
3. <http://openoffice.mirahost.ru> – сайт, где выложен скрипт и образцы файлов настроек.