Guide to rsnapshot and incremental backups on Linux

Introduction

rsnapshot is a backup tool written in Perl that utilizes rsync as its back-end. rsnapshot allows users to create customized incremental backup solutions. This article will discuss the following: the benefits of an incremental backup solution, rsnapshot's installation, its configuration, and usage examples.

Back-it up!

I was recently discussing with a colleague the benefits of backing up your data. My colleague was telling me how one of her customers had recently lost a rather lengthy article that they had been working on. I decided that this may be a good chance to experiment with my netbook and rsnapshot. For this tutorial, I'll assume you have have 2 pieces of hardware: your host computer, and your destination equipment. I'll be using an external hard drive for the majority of this post. However, I will briefly cover usage for backing up files over a LAN.

Backing up your data should not be the question to ask but rather how should I backup my stuff? What's the best way? Well there are many different backup pathways you can take, including block level (dd, partimage), partition level (RAID and all its variations), file level (rsyncand its children applications). I'll discuss two types of backups in the context of file-based backups.

Normal backups, or full backups, are self explanatory. Normal backups are one way of backing up ALL your files every time you perform a backup. One issue with utilizing a multiple normal backup scheme is that a normal backup takes up a considerable amount of space. For example, if you perform a full backup of a 250gig hard drive at 20% capacity, everyday for just one week (assuming that the amount of data does not fluctuate) will mean that you already have used 350gigs for only one week's worth of backups. As you can see, that is not feasible in the long run. The other method that I prefer is the incremental backup method. An incremental backup consists of one full backup and then performing additional backups. These additional backups will **only** backup files that have changed since the last backup. Instead of backing up your entire hard drive, only the specific files that have changed since the last backup are backed up. As you can probably imagine this is a much more efficient process. One tool that does this on *nix is rsnapshot.

What is rsnapshot?

rsnapshot, as mentioned earlier, is an incremental backup utility. In this tutorial, I will show you how to create a seven day rotation backup scheme using rsnapshot. Essentially, rsnapshot will create 1 full backup and then subsequent backups will backup only files that have changed. The true power of rsnapshot is its ability to utilize hard-links between each backup. Each backup will appear to be a full backup. In reality each new backup consists of newly created or updated files. rsnapshot can be used over a LAN and can also be ran from cron. In this tutorial, I'll show both usage examples.

Installation

Installation of rsnapshot is pretty straightforward, simply run the following linux command:

On Debian (or Ubuntu):

```
apt-get install rsnapshot
```

On Fedora:

```
yum install rsnapshot
```

On ArchLinux:

```
pacman -S rsnapshot
```

Now let's configure rsnapshot.

Configuration

Take a look at /etc, if /etc/rsnapshot.conf.default exists you need to copy it to /etc/rsnapshot.conf. If the .default file does not exist, then create a backup of the regular .conf. This is useful in the case you need to reference it later on.

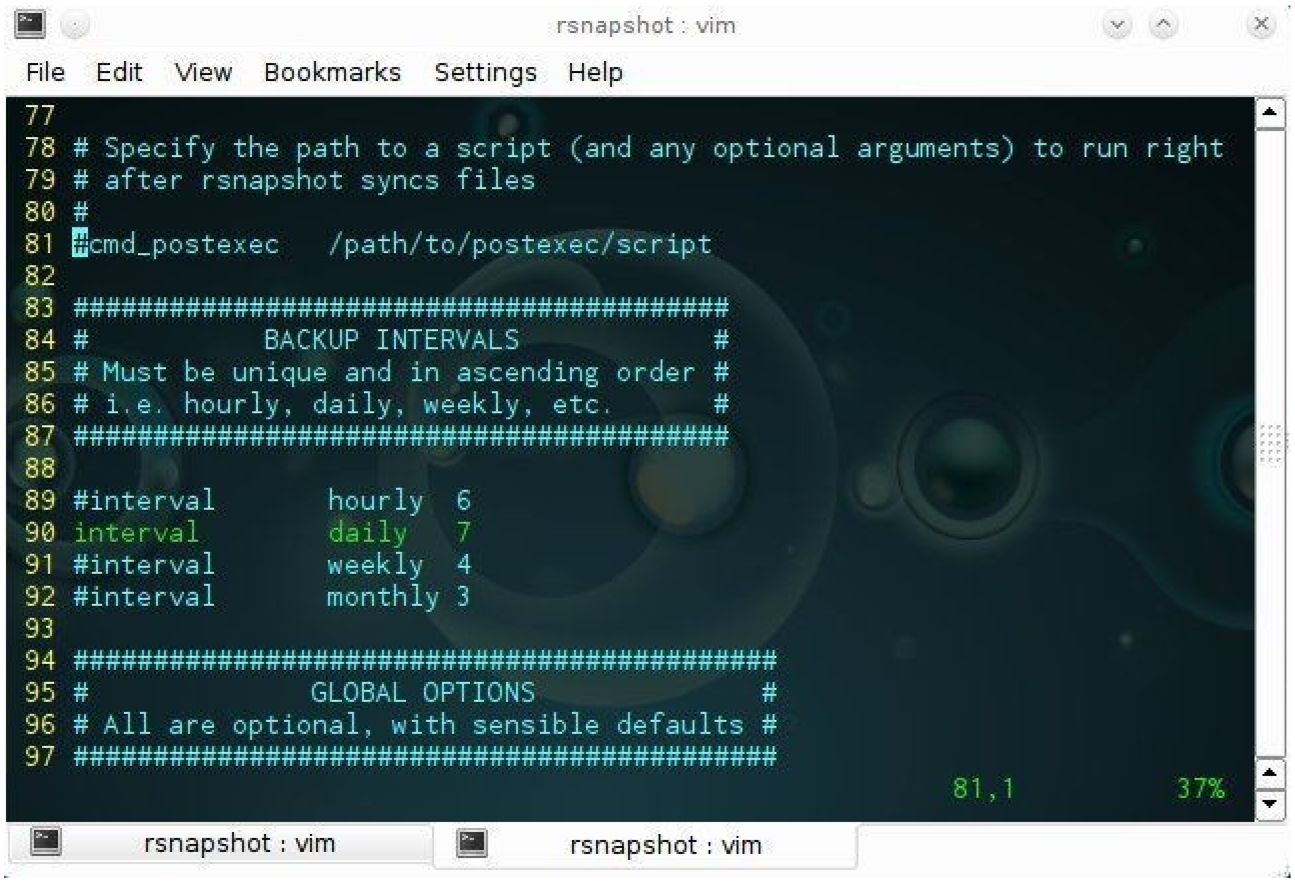
Open up rsnapshot.conf and start editing it to your needs. rsnapshot's configuration file is well commented. This makes configuring it much easier. I would start by uncommenting all the applications within the "External Program Dependencies" section of the configuration file. For starters, uncomment the following lines:

```
cmd_rsync /USO/bin/rsync
cmd_d /bin/D
cmd_rsnapshot_diff /USA/bin/rsnapshot-diff
```

You may need to change the location of rsnapshot-diff, if rsnapshot doesn't locate it. To make sure you are referencing the correct binary issue:

```
whereis rsnapshot-diff
```

Now you need to decide how exactly to design your backup scheme. Once you have settled on a scheme, you need to edit the “interval” lines located under “Backup Intervals”. rsnapshot can do hourly, daily, weekly, and monthly backups. My system does a seven day backup scheme, but you can decide here what you would like to do. One example could be creating a backup that runs every six hours, everyday of the week. This is up to you. I’ll use my setup as an example. See the screenshot below:



```

# Specify the path to a script (and any optional arguments) to run right
# after rsnapshot syncs files
#
#cmd_postexec   /path/to/postexec/script

###########################################
#            BACKUP INTERVALS             #
# Must be unique and in ascending order #
# i.e. hourly, daily, weekly, etc.      #
###########################################

#interval       hourly  6
interval        daily   7
#interval       weekly  4
#interval       monthly 3

##############################################
#              GLOBAL OPTIONS                #
# All are optional, with sensible defaults #
##############################################
                                                        81,1          37%
```

The other two lines that you need to uncomment are the snapshot_root (located at the beginning of the script) and backup (located under “BACKUP POINTS/SCRIPTS”).

```
snapshot_root $destination/ #specifies where your backups are stored
.
.
backup $source/ $hastens
#backup specifies what you what to backup, you can backup
#from multiple locations just specify multiple backup lines
```

The backup line consists of three parts. These three parts include the declaration "backup", the source location, and the hastened. Each component MUST be separated by a tab and directories must end with / or rsnapshot will not properly. Two more features worth noting is the logging functionality and the ability to use rsync's include/exclude files. Uncomment the following three lines (and amend accordingly):

```
verbose 5 #How much information should the
 #actual backup relay to you?
loglevel 5 #How must information
 #should be stored in the log file?
logfile $log_file_destination #Where should the logfile be stored?
```

As you probably noticed from reading the comments in the .conf file, verbosity ranges from 1-5, with5 being most verbose. I would enable this at first to see if everything is running okay. This level of control is helpful. If you run into any issues, you have the ability to perform some debugging.

As mentioned earlier, you can also use rsync include and exclude files and below is an example exclude file.

```
#EXCLUDE
#not to backup alongside ~
- /home/javier/data/
- /home/javier/$dest/
```

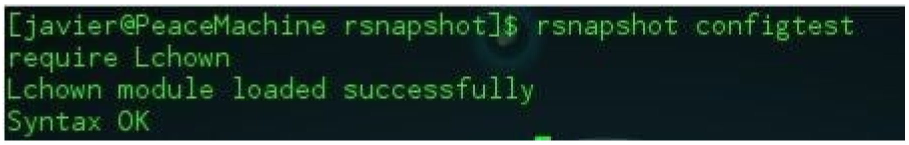
Include files are similar in nature. Instead of '-' use '+' to denote data that you want to include. If you decide that you would like to use include or exclude files take a look at the include_file/exclude_file lines. Here is my section of the rsnapshot.conf noting these options.

```
include_file /home/javier/backup/include.lst
exclude_file /home/javier/backup/exclude.lst
```

If you have been following along up to this point rsnapshot is pretty much configured. If you want to run anything before or after rsnapshot finishes take a look at the cmd_preexec and cmd_postexec lines. You can uncomment these lines and specify scripts to run prior to or after the completion of a backup if needed. If not, you're finished editing your config file. Now run the following linux command:

```
rsnapshot configtest
```

Screenshot below:



```
[javier@PeaceMachine rsnapshot]$ rsnapshot configtest
require Lchown
Lchown module loaded successfully
Syntax OK
```

That command runs rsnapshot and tells it to perform sanity check on rsnapshot.conf. If everything went okay, then you should get the following output: "Syntax OK." If not read the output it gives you and edit rsnapshot.conf as needed.

Usage Examples

I'll discuss three forms of using rsnapshot. These forms include locally, remotely, and its automation (through cron).Usage doesn't necessary differ between local or remote execution. Rather, I'll point out the differences in configuration files

Using rsnapshot Locally

For local backup the two important lines are:

```
snapshot_root $dest/
backup $source/ $hostname/
```

You can specify multiple backup sources, by just creating multiple lines referring to each directory you want to backup.
To actually run a backup issue the following linux command:

```
rsnapshot daily
```

You can also use hourly, weekly, and monthly as options. Each command will create a backup directory according to the specified operation. If I browse to my backup location, I will see the following:

```
[some_user@hostname backupdir]# ls
daily.0 daily.1 log
```

The output above shows that I have two backups of a my seven day backup scheme created.

Remote Backup

To utilize rsnapshot's remote backup capabilities simply specify the remote location as your backup source (To enable remote backup, ssh is going to have to be enabled for the server. Key based authentication will be your best option here):

```
backup root@example.com:/home/ example.com/
```

Once you have specified a remote source as a location to backup simply run rsnapshot. Example below:

```
rsnapshot hourly
```

Automation with Cron

If you have any experience with cron then adding an entry with snapshot merely consists of adding the specific command e.g. "rsnapshot hourly" and the relevant syntax on cron. Here are some examples:

```
20 23 * * * /usr/bin/rsnapshot daily # daily backup is ran at 11:20 pm
05 23 * * 7 /usr/bin/rsnapshot weekly # weekly backup is ran at 11:05 pm
 # on Sunday
```

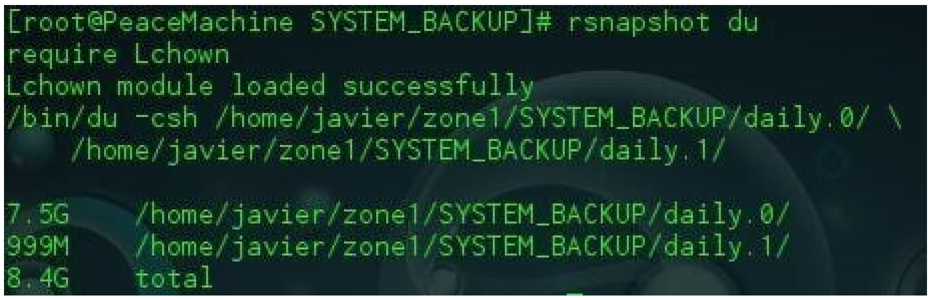
It is recommended that you schedule larger backups before smaller backups (as shown above) to prevent conflicts with rotations between backups. In addition, it is recommended to create a gap between the larger backups and smaller backups to avoid conflicts between each job.

Other uses of rsnapshot

rsnapshot comes with several other useful features. For example, you can run the following linux command:

```
rsnapshot du
```

to see how much disk space it is using (du must be uncommented in /etc/rsnapshot.conf). Screenshot below:



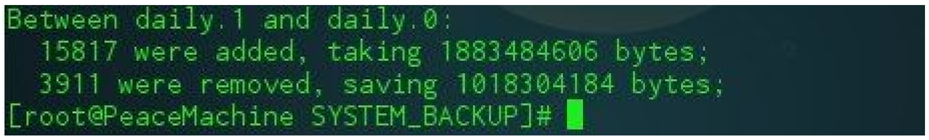
```
[root@PeaceMachine SYSTEM_BACKUP]# rsnapshot du
require Lchown
Lchown module loaded successfully
/bin/du -csh /home/javier/zone1/SYSTEM_BACKUP/daily.0/ \
    /home/javier/zone1/SYSTEM_BACKUP/daily.1/

7.5G    /home/javier/zone1/SYSTEM_BACKUP/daily.0/
999M    /home/javier/zone1/SYSTEM_BACKUP/daily.1/
8.4G    total
```

You can run the following to compare the changes between backups:

```
rsnapshot diff daily.0 daily1
```

You should see output similar to the screenshot below:



```
Between daily.1 and daily.0:
  15817 were added, taking 1883484606 bytes;
  3911 were removed, saving 1018304184 bytes;
[root@PeaceMachine SYSTEM_BACKUP]#
```

See the rsnapshot's man page for additional functionality.

Troubleshooting

If you get any issues about Lchown run the following linux command:

```
perl -MCPAN -e 'install QC(Lchown)'
```

Conclusion

After successfully completing this tutorial you should now have a high quality backup scheme in place. For additional documentation check out rsnapshot's man page, and its homepage located here. It contains an excellent how-to, which is available in several formats. Additionally, I would recommend checking out other backup programs, these include rsync (back end to rsnapshot), rdiff-backup, partimage, and dd.