

## 骏马金龙

网名骏马金龙，钟情于IT世界里的各种原理和实现机制，强迫症重症患者。爱研究、爱翻译、爱分享。特借此一亩三分田记录自己成长点滴！！！

博客园

首页

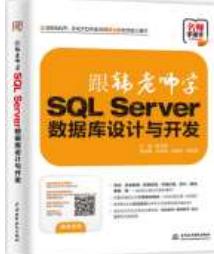
新随笔

管理

随笔 - 161 文章 - 2 评论 - 121



我的作品：



系列文章目录：

- 1. [Linux回炉复习系列](#)
  - 2. [数据库系列](#)
  - 3. [网站架构从LAMP开始](#)
- 昵称：骏马金龙  
园龄：2年8个月  
粉丝：123  
关注：3  
+加关注

2017年12月						
日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

随笔分类(249)

[java SE\(4\)](#)[Linux 基础篇\(57\)](#)[Linux 杂项\(61\)](#)[Linux服务篇\(32\)](#)[OpenSSL\(19\)](#)[shell\(37\)](#)[sql server\(2\)](#)[vmware\(1\)](#)[数据库系列\(15\)](#)[网站架构\(21\)](#)

## 第2章 rsync(一)：基本命令和用法

以下是rsync系列篇：

1. [rsync\(一\)：基本命令和用法](#)
2. [rsync\(二\)：inotify+rsync详细说明和sersync](#)
3. [rsync算法原理和工作流程分析](#)
4. [rsync技术报告\(翻译\)](#)
5. [rsync工作机制\(翻译\)](#)
6. [man rsync翻译\(rsync命令中文手册\)](#)

## 本文目录：

- [\*\*2.1 说在前面的话\*\*](#)
- [\*\*2.2 rsync同步基本说明\*\*](#)
- [\*\*2.3 rsync三种工作方式\*\*](#)
- [\*\*2.4 选项说明和示例\*\*](#)
- [\*\*2.4.1 基础示例\*\*](#)
- [\*\*2.4.2 "--exclude"排除规则\*\*](#)
- [\*\*2.4.3 "--delete"的解释\*\*](#)
- [\*\*2.5 rsync daemon模式\*\*](#)
- [\*\*2.5.1 简单介绍\*\*](#)
- [\*\*2.5.2 daemon配置文件rsyncd.conf\*\*](#)

## [\*\*2.6 远程shell方式连接使用daemon\*\*](#)

### **2.1 说在前面的话**

rsync官方网站: <https://www.samba.org/ftp/rsync/rsync.html>

rsync是可以实现增量备份的工具。配合任务计划，rsync能实现定时或间隔同步，配合inotify或sersync，可以实现触发式的实时同步。

rsync可以实现scp的远程拷贝(rsync不支持远程到远程的拷贝，但scp支持)、cp的本地拷贝、rm删除和"ls -l"显示文件列表等功能。但需要注意的是，rsync的最终目的或者说其原始目的是实现两端主机的文件同步，因此实现的scp/cp/rm等功能仅仅只是同步的辅助手段，且rsync实现这些功能的方式和这些命令是不一样的。事实上，rsync有一套自己的算法，其算法原理以及rsync对算法实现的机制可能比想象中要复杂一些。平时使用rsync实现简单的备份、同步等功能足以，没有多大必要去深究这些原理性的内容。但是想要看懂rsync命令的man文档、使用"-vvvv"分析rsync执行过程，以及实现rsync更强大更完整的功能，没有这些理论知识的支持是绝对不可能实现的。本篇文章将简单介绍rsync的使用方法和它常用的功能。在本篇文章之后的下几篇文章中，将介绍inotify+rsync和sersync，再之后将详细解释rsync相关的原理，其中包括官方技术报告的翻译(即算法原理)、rsync同步的整个过程(也是官方推荐文章的翻译)，然后专门使用一篇文章通过示例来详细解释rsync算法原理，最后给出rsync的man文档翻译。希望各位朋友能藉此深入rsync。

## 随笔档案(161)

2017年11月 (3)

2017年10月 (23)

2017年9月 (36)

2017年8月 (38)

2017年7月 (15)

2017年6月 (24)

2017年2月 (2)

2016年11月 (12)

2016年10月 (1)

2016年9月 (4)

2016年8月 (2)

2016年3月 (1)

## 积分与排名

积分 - 71424

排名 - 4479

## 最新评论

1. Re:Linux find运行机制详解

@徐增强感谢支持...

--骏马金龙

2. Re:Linux find运行机制详解

这么好的文章竟然没人看

--徐增强

3. Re:Ansible系列(二) : 选项和常用模块

@徐增强多谢支持...

--骏马金龙

4. Re:Ansible系列(二) : 选项和常用模块

很好基础适合我这样的小白新手已关注

--徐增强

5. Re:mysql、mariadb安装和多实例配置

回归正题，以下是rsync相关基础内容。

## 2.2 rsync同步基本说明

rsync的目的是实现本地主机和远程主机上的文件同步(包括本地推到远程，远程拉到本地两种同步方式)，也可以实现本地不同路径下文件的同步，但不能实现远程路径1到远程路径2之间的同步(scp可以实现)。

不考虑rsync的实现细节，就文件同步而言，涉及了源文件和目标文件的概念，还涉及了以哪边文件为同步基准。例如，想让目标主机上的文件和本地文件保持同步，则是以本地文件为同步基准，将本地文件作为源文件推送到目标主机上。反之，如果想让本地主机上的文件和目标主机上的文件保持同步，则目标主机上的文件为同步基准，实现方式是将目标主机上的文件作为源文件拉取到本地。当然，要保持本地的两个文件相互同步，rsync也一样能实现，这就像Linux中cp命令一样，以本地某文件作为源，另一文件作为目标文件，但请注意，虽然rsync和cp能达到相同的目的，但它们的实现方式是不一样的。

既然是文件同步，在同步过程中必然会涉及到源和目标两文件之间版本控制的问题，例如是否要删除源主机上没有但目标上多出来的文件，目标文件比源文件更新(newer than source)时是否仍要保持同步，遇到软链接时是拷贝软链接本身还是拷贝软链接所指向的文件，目标文件已存在时是否要先对其做个备份等等。

**rsync同步过程中由两部分模式组成：决定哪些文件需要同步的检查模式以及文件同步时的同步模式。**

(1). 检查模式是指按照指定规则来检查哪些文件需要被同步，例如哪些文件是明确被排除不传输的。**默认情况下，rsync使用“quick check”算法快速检查源文件和目标文件的大小、mtime(修改时间)是否一致，如果不一致则需要传输。**当然，也可以通过在rsync命令行中指定某些选项来改变quick check的检查模式，比如"--size-only"选项表示"quick check"将仅检查文件大小不同的文件作为待传输文件。rsync支持非常多的选项，其中检查模式的自定义性是非常有弹性的。

(2). 同步模式是指在文件确定要被同步后，在同步过程发生之前要做哪些额外工作。例如上文所说的是是否要先删除源主机上没有但目标主机上有的文件，是否要先备份已存在的目标文件，是否要追踪链接文件等额外操作。rsync也提供非常多的选项使得同步模式变得更具弹性。

相对来说，为rsync手动指定同步模式的选项更常见一些，只有在有特殊需求时才指定检查模式，因为大多数检查模式选项都可能会影响rsync的性能。

## 2.3 rsync三种工作方式

以下是rsync的语法：

```
Local: rsync [OPTION...] SRC... [DEST]
Access via remote shell:
  Pull: rsync [OPTION...] [USER@]HOST:SRC... [DEST]
  Push: rsync [OPTION...] SRC... [USER@]HOST:DEST
Access via rsync daemon:
  Pull: rsync [OPTION...] [USER@]HOST::SRC... [DEST]
        rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
  Push: rsync [OPTION...] SRC... [USER@]HOST::DEST
        rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
```

由此语法可知，rsync有三种工作方式：

(1). 本地文件系统上实现同步。命令行语法格式为上述"Local"段的格式。

(2). 本地主机使用远程shell和远程主机通信。命令行语法格式为上述"Access via remote shell"段的格式。

(3). 本地主机通过网络套接字连接远程主机上的rsync daemon。命令行语法格式为上述"Access via rsync daemon"段的格式。

前两者的本质是通过管道通信，即使是远程shell。而方式(3)则是让远程主机上运行rsync服务，使其监听在一个端口上，等待客户端的连接。

但是，**通过远程shell也能临时启动一个rsync daemon，这不同于方式(3)，它不要求远程主机上事先启动rsync服务，而是临时派生出rsync daemon，它是单用途的一次性daemon**，仅用于临时读取daemon的配置文件，当此次rsync同步完成，远程shell启动的rsync daemon进程也会自动消逝。此通信方式的命令行语法格式同"Access via rsync daemon"，但要求options部分必须明确指定"--rsh"选项或其短选项"-e"。

以下是对rsync语法的简单说明，由于rsync支持一百多个选项，所以此处只介绍几个常用选项。完整的选项说明以及rsync的使用方法见我翻译的"[man rsync](#)"。

```
Local: rsync [OPTION...] SRC... [DEST]
Access via remote shell:
  Pull: rsync [OPTION...] [USER@]HOST:SRC... [DEST]
  Push: rsync [OPTION...] SRC... [USER@]HOST:DEST
```

不改也可以，改它只是出于安全考虑。mysqld以mysql用户身份运行，将/usr/local/mysql改回root可以chroot mysql相关的进程。

--骏马金龙

## 阅读排行榜

1. xargs的原理剖析及用法详解(703)  
6)

2. Linux回炉复习系列文章总目录(57)  
61)

3. OpenSSL主配置文件openssl.cnf  
(3888)

4. shell脚本--echo和printf打印输出  
(2295)

5. grub2详解(翻译和整理官方手册)  
(1727)

6. SHELL脚本--多命令逻辑执行顺序  
(1624)

7. SHELL脚本--expr命令全解(1536)

8. 第1章 文件类基础命令(1466)

9. 第2章 rsync(一) : 基本命令和用法  
(1396)

10. openssl签署和自签署证书的多种  
实现方式(1207)

11. rsync算法原理和工作流程分析(1)  
178)

12. OpenSSL命令系列(1154)

13. 加密、签名和SSL握手机制细节(1)  
138)

14. 第14章 Linux开机详细流程(103)  
0)

15. 第4章 ext文件系统机制(1023)

16. 第13章 Linux的网络管理(800)

17. PXE+kickstart无人值守安装Cen  
tos 6(800)

18. shell解析命令行的过程以及eval  
命令(759)

19. 文本排序的王者：玩透sort命令(7)  
48)

20. nginx的反向代理功能和缓存功能  
(746)

```
Access via rsync daemon:
Pull: rsync [OPTION...] [USER@]HOST::SRC... [DEST]
      rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
Push: rsync [OPTION...] SRC... [USER@]HOST::DEST
      rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
```

其中，第一个路径参数一定是源文件路径，即作为同步基准的一方，可以同时指定多个源文件路径。最后一个路径参数则是目标文件路径，也就是待同步方。路径的格式可以是本地路径，也可以是使用user@host:path或user@host::path的远程路径，如果主机和path路径之间使用单个冒号隔开，表示使用的是远程shell通信方式，而使用双冒号隔开的则表示的是连接rsync daemon。另外，连接rsync daemon时，还提供了URL格式的路径表达方式rsync://user@host/path。

**如果仅有一个SRC或DEST参数，则将以类似于"ls -l"的方式列出源文件列表(只有一个路径参数，总会认为是源文件)，而不是复制文件。**

如果对rsync不熟悉，可暂先只了解本地以及远程shell格式的user@host:path路径格式。例如：

[root@xuexi ~]# rsync /etc/fstab /tmp	# 在本地同步
[root@xuexi ~]# rsync -r /etc 172.16.10.5:/tmp	# 将本地/etc目录拷贝到远程主机的/tmp下， 以保证远程/tmp目录和本地/etc保持同步
[root@xuexi ~]# rsync -r 172.16.10.5:/etc /tmp	# 将远程主机的/etc目录拷贝到本地/tmp下， 以保证本地/tmp目录和远程/etc保持同步
[root@xuexi ~]# rsync /etc/	# 列出本地/etc目录下的文件列表
[root@xuexi ~]# rsync 172.16.10.5:/tmp/	# 列出远程主机上/tmp/目录下的文件列表

另外，使用rsync一定要注意的一点是，**源路径如果是一个目录的话，带上尾随斜线和不带尾随斜线是不一样的，不带尾随斜线表示的是整个目录包括目录本身，带上尾随斜线表示的是目录中的文件，不包括目录本身**。例如：

```
[root@xuexi ~]# rsync /etc /tmp
[root@xuexi ~]# rsync /etc/ /tmp
```

第一个命令会在/tmp目录下创建etc目录，而第二个命令不会在/tmp目录下创建etc目录，源路径/etc/中的所有文件都直接放在/tmp目录下。

## 2.4 选项说明和示例

接下来是rsync的选项说明。

-v : 显示rsync过程中详细信息。可以使用"-vvvv"获取更详细信息。  
- P : 显示文件传输的进度信息。(实际上"-P"="--partial --progress"，其中的"--progress"才是显示进度信息的)。  
- n --dry-run : 仅测试传输，而不实际传输。常和"-vvvv"配合使用来查看rsync是如何工作的。  
- a --archive : 归档模式，表示递归传输并保持文件属性。等同于"-rtopgDl"。  
- r --recursive : 递归到目录中去。  
- t --times : 保持mtime属性。**强烈建议任何时候都加上"-t"，否则目标文件mtime会设置为系统时间，导致下次更新**  
**新**  
**：检查出mtime不同从而导致增量传输无效。**  
- o --owner : 保持owner属性(属主)。  
- g --group : 保持group属性(属组)。  
- p --perms : 保持perms属性(权限，不包括特殊权限)。  
- D : 是"--device --specials"选项的组合，即也拷贝设备文件和特殊文件。  
- l --links : 如果文件是软链接文件，则拷贝软链接本身而非软链接所指向的对象。  
- z : 传输时进行压缩提高效率。  
- R --relative : 使用相对路径。意味着将命令行中指定的全路径而非路径最尾部的文件名发送给服务端，包括它们的属性。用法见下文示例。  
--size-only : 默认算法是检查文件大小和mtime不同的文件，使用此选项将只检查文件大小。  
- u --update : 仅在源mtime比目标已存在文件的mtime新时才拷贝。注意，该选项是接收端判断的，不会影响删除行为。  
- d --dirs : 以不递归的方式拷贝目录本身。默认递归时，如果源为"dir1/file1"，则不会拷贝dir1目录，使用该选项将拷贝dir1但不拷贝file1。  
--max-size : 限制rsync传输的最大文件大小。可以使用单位后缀，还可以是一个小数值(例如："--max-size=1.5m")。  
--min-size : 限制rsync传输的最小文件大小。这可以用于禁止传输小文件或那些垃圾文件。  
--exclude : 指定排除规则来排除不需要传输的文件。  
--delete : 以SRC为主，对DEST进行同步。多则删之，少则补之。注意"--delete"是在接收端执行的，所以它是在  
**：exclude/include规则生效之后才执行的。**  
- b --backup : 对目标上已存在的文件做一个备份，备份的文件名后默认使用"~"做后缀。  
--backup-dir : 指定备份文件的保存路径。不指定时默认和待备份文件保存在同一目录下。  
- e : 指定所要使用的远程shell程序，默认为ssh。  
- port : 连接daemon时使用的端口号，默认为873端口。  
--password-file : daemon模式时的密码文件，可以从中读取密码实现非交互式。注意，这不是远程shell认证的密码，而是rsync模块认证的密码。  
- w --whole-file : rsync将不再使用增量传输，而是全量传输。在网络带宽高于磁盘带宽时，该选项比增量传输更高效。  
--existing : 要求只更新目标端已存在的文件，目标端还不存在的文件不传输。注意，使用相对路径时如果上层目录不存在也不会传输。

## 推荐排行榜

1. Linux回炉复习系列文章总目录(7)
2. bash启动时加载配置文件过程(7)
3. 文本排序的王者：玩透sort命令(7)
4. sed修炼系列(二) : sed武功心法(in fo sed翻译+注解)(5)
5. 加密、签名和SSL握手机制细节(4)
6. xargs的原理剖析及用法详解(4)
7. shell解析命令行的过程以及eval命令(3)
8. String、StringBuilder和StringBuffer类(3)
9. nginx的反向代理功能和缓存功能(3)
10. nginx作为web服务以及nginx.conf详解(3)
11. grep命令中文手册(info grep翻译)(3)
12. 详细分析apache httpd反向代理的用法(3)
13. 五种IO模型分析(3)
14. 零复制(zero copy)技术(2)
15. MySQL基本语法(一) : 和SQL Server语法的差异小归纳(2)
16. Ansible系列(二) : 选项和常用模块(2)
17. awk知识点全回顾(2)
18. sed修炼系列(四) : sed中的疑难杂症(2)
19. sed修炼系列(一) : 花拳绣腿之入门篇(2)
20. (MariaDB)MySQL内置函数大全(2)

--ignore-existing : 要求只更新目标端不存在的文件。和"--existing"结合使用有特殊功能，见下文示例。  
--remove-source-files : 要求删除源端已经成功传输的文件。

rsync的选项非常多，能够实现非常具有弹性的功能，以上选项仅仅只是很小一部分常用的选项，关于更完整更详细的选项说明，见我的rsync man手册翻译。

虽然选项非常多，但最常用的选项组合是"avz"，即压缩和显示部分信息，并以归档模式传输。

### 2.4.1 基础示例

以下几个本地同步示例和通过远程shell实现的同步示例，示例中没有使用"-a"选项，目的是为了更清晰地说明各选项的作用。

#### (1).将/etc/fstab拷贝到/tmp目录下。

```
[root@xuexi ~]# rsync /etc/fstab /tmp
```

#### (2).将/etc/cron.d目录拷贝到/tmp下。

```
[root@xuexi ~]# rsync -r /etc/cron.d /tmp
```

该命令会在目标主机上创建/tmp/cron.d目录，并将/etc/cron.d/中的文件放入到/tmp/cron.d/目录中，也就是说默认情况下，是不会在目录路径下创建上层目录/etc的。

#### (3).将/etc/cron.d目录拷贝到/tmp下，但要求在/tmp下也生成etc子目录。

```
[root@xuexi ~]# rsync -R -r /etc/cron.d /tmp
```

其中"-R"选项表示使用相对路径，此相对路径是以目标目录为根的。对于上面的示例，表示在目标上的/tmp下创建etc/cron.d目录，即/tmp/etc/cron.d，etc/cron.d的根"/"代表的就是目标/tmp。

如果要拷贝的源路径较长，但只想在目标主机上保留一部分目录结构，例如要拷贝/var/log/anaconda/\*到/tmp下，但只想在/tmp下保留从log开始的目录，如何操作？使用一个点代表相对路径的起始位置即可，也就是将长目录进行划分。

```
[root@xuexi ~]# rsync -R -r ./var/.log/anaconda /tmp
```

这样，从点开始的目录都是相对路径，其相对根目录为目标路径。所以对于上面的示例，将在目标上创建/tmp/log/anaconda/\*。

#### (4).对远程目录下已存在文件做一个备份。

```
[root@xuexi ~]# rsync -R -r --backup /var/.log/anaconda /tmp
```

这样在目标目录下，已存在的文件就被做一个备份，备份文件默认使用"~"做后缀，可以使用"--suffix"指定备份后缀。

```
[root@xuexi tmp]# ll log/anaconda/
total 3112
-rw----- 1 root root    6668 Jul 14 12:45 anaconda.log
-rw----- 1 root root    6668 Jul 14 11:44 anaconda.log~
-rw----- 1 root root   3826 Jul 14 12:45 ifcfg.log
-rw----- 1 root root   3826 Jul 14 11:44 ifcfg.log~
-rw----- 1 root root 1102699 Jul 14 12:45 journal.log
-rw----- 1 root root 1102699 Jul 14 11:44 journal.log~
-rw----- 1 root root      0 Jul 14 12:45 ks-script-luLekR.log
-rw----- 1 root root      0 Jul 14 11:44 ks-script-luLekR.log~
-rw----- 1 root root      0 Jul 14 12:45 ks-script-iGp14q.log
-rw----- 1 root root      0 Jul 14 11:44 ks-script-iGp14q.log~
-rw----- 1 root root 160420 Jul 14 12:45 packaging.log
-rw----- 1 root root 160420 Jul 14 11:44 packaging.log~
-rw----- 1 root root  27906 Jul 14 12:45 program.log
-rw----- 1 root root  27906 Jul 14 11:44 program.log~
-rw----- 1 root root  78001 Jul 14 12:45 storage.log
-rw----- 1 root root  78001 Jul 14 11:44 storage.log~
-rw----- 1 root root 197961 Jul 14 12:45 syslog
-rw----- 1 root root 197961 Jul 14 11:44 syslog~
```

可以使用"--backup-dir"指定备份文件保存路径，但要求保存路径必须存在。

```
[root@xuexi ~]# mkdir /tmp/log_back
```

```
[root@xuexi ~]# rsync -R -r --backup --backup-dir=/tmp/log_back /var/.log/anaconda /tmp
```

指定备份路径后，默认将不会加备份后缀，除非使用"--suffix"显式指定后缀，如"--suffix=~"。

```
[root@xuexi tmp]# tree /tmp/log_back/
/tmp/log_back/
└── log
    └── anaconda
```

```

├── anaconda.log
├── ifcfg.log
├── journal.log
├── ks-script-luLekR.log
├── ks-script-iGpl4q.log
├── packaging.log
├── program.log
└── storage.log
└── syslog

```

**(5).指定ssh连接参数，如端口、连接的用户、ssh选项等。**

```

[root@xuexi tmp]# >~/.ssh/known_hosts    # 先清空host key以便下面的测试

[root@xuexi tmp]# rsync -e "ssh -p 22 -o StrictHostKeyChecking=no" /etc/fstab
172.16.10.5:/tmp
Warning: Permanently added '172.16.10.5' (RSA) to the list of known hosts.
root@172.16.10.5's password:

```

可见直接指定ssh参数是生效的。

**(6)."--existing"和"--ignore-existing"**

"--existing"是只更新目标端已存在的文件。

目前/tmp/{a,b}目录中内容如下，bashrc在a目录中，crontab在b目录中，且a目录中多了一个c子目录。

```

[root@xuexi ~]# tree /tmp/{a,b}
/tmp/a
├── bashrc
├── c
│   └── find
├── fstab
├── profile
└── rc.local
/tmp/b
├── crontab
├── fstab
└── profile
└── rc.local

1 directory, 9 files

```

使用"--existing"选项使得只更新目标端已存在的文件。

```

[root@xuexi ~]# rsync -r -v --existing /tmp/a/ /tmp/b
sending incremental file list
fstab
profile
rc.local

sent 2972 bytes received 70 bytes 6084.00 bytes/sec
total size is 204755 speedup is 67.31

```

结果只有3个目标上已存在的文件被更新了，由于目标上没有c目录，所以c目录中的文件也没有进行传输。

而"--ignore-existing"是更新目标端不存在的文件。

```

[root@xuexi ~]# rsync -r -v --ignore-existing /tmp/a/ /tmp/b
sending incremental file list
bashrc
c/
c/find

sent 202271 bytes received 54 bytes 404650.00 bytes/sec
total size is 204755 speedup is 1.01

```

**(7)."--remove-source-files"删除源端文件。**

使用该选项后，源端已经更新成功的文件都会被删除，源端所有未传输或未传输成功的文件都不会被移除。未传输成功的原因有多种，如exclude排除了，"quick check"未选项该文件，传输中断等等。

总之，显示在"rsync -v"被传输列表中的文件都会被移除。如下：

```

[root@xuexi ~]# rsync -r -v --remove-source-files /tmp/a/anaconda /tmp/a/audit /tmp
sending incremental file list
anaconda/anaconda.log
anaconda/ifcfg.log

```

```

anaconda/journal.log
anaconda/ks-script-luLekR.log
anaconda/ks-script-iGpl4q.log
anaconda/packaging.log
anaconda/program.log
anaconda/storage.log
anaconda/syslog
audit/audit.log

sent 4806915 bytes received 204 bytes 9614238.00 bytes/sec
total size is 4805676 speedup is 1.00

```

上述显示出来的文件在源端全部被删除。

## 2.4.2 "--exclude"排除规则

使用"--exclude"选项指定排除规则，排除那些不需要传输的文件。

```

[root@xuexi tmp]# rsync -r -v --exclude="anaconda/*.log" /var/log/anaconda
/var/log/audit /tmp
sending incremental file list
anaconda/
anaconda/syslog
audit/
audit/audit.log

sent 3365629 bytes received 58 bytes 6731374.00 bytes/sec
total size is 3365016 speedup is 1.00

```

上例中只排除了anaconda目录中的log文件，但是audit目录中的log文件是正常传输的。

注意，一个"--exclude"只能指定一条规则，要指定多条排除规则，需要使用多个"--exclude"选项，或者将排除规则写入到文件中，然后使用"--exclude-from"选项读取该规则文件。

另外，除了"--exclude"排除规则，还有"--include"包含规则，顾名思义，它就是筛选出要进行传输的文件，所以include规则也称为传输规则。它的使用方法和"--exclude"一样。如果一个文件即能匹配排除规则，又能匹配包含规则，则先匹配到的立即生效，生效后就不再进行任何匹配。

最后，关于规则，最重要的一点是它的作用时间。**当发送端敲出rsync命令后，rsync将立即扫描命令行中给定的文件和目录(扫描过程中还会按照目录进行排序，将同一个目录的文件放在相邻的位置)，这称为拷贝树(copy tree)，扫描完成后将待传输的文件或目录记录到文件列表中，然后将文件列表传输给接收端。而筛选规则的作用时刻是在扫描拷贝树时，所以会根据规则来匹配并决定文件是否记录到文件列表中(严格地说是会记录到文件列表中的，只不过排除的文件会被标记为hide隐藏起来)，只有记录到了文件列表中的文件或目录才是真正需要传输的内容。换句话说，筛选规则的生效时间在rsync整个同步过程中是非常靠前的，它会影响很多选项的操作对象，最典型的如"--delete"。**也许，你看完这一整篇文章都没感觉到这一点的重要性，但如果你阅读rsync的man文档或者学习rsync的原理，你一定会深有体会。

实际上，排除规则和包含规则都只是"--filter"筛选规则的两种特殊规则。"--filter"比较复杂，它有自己的规则语法和匹配模式，由于篇幅有限，以及考虑到本文的难度定位，"--filter"规则不便在此多做解释，仅简单说明下规则类，帮助理解下文的"--delete"。

以下是rsync中的规则种类，不解之处请结合下文的"--delete"分析：

(1).exclude规则：即排除规则，只作用于发送端，被排除的文件不会进入文件列表(实际上是加上隐藏规则进行隐藏)。

(2).include规则：即包含规则，也称为传输规则，只作用于发送端，被包含的文件将明确记录到文件列表中。

(3).hide规则：即隐藏规则，只作用于发送端，隐藏后的文件对于接收端来说是看不见的，也就是说接收端会认为它不存在于源端。

(4).show规则：即显示规则，只作用于发送端，是隐藏规则的反向规则。

(5).protect规则：即保护规则，该规则只作用于接收端，被保护的文件不会被删除掉。

(6).risk规则：即取消保护规则。是protect的反向规则。

除此之外，还有一种规则是"clear规则"，作用是删除include/exclude规则列表。

## 2.4.3 "--delete"解释

使用"--delete"选项后，接收端的rsync会先删除目标目录下已经存在，但源端目录不存在的文件。也就是"多则删之，少则补之"。

例如，先实现一次同步，再向目标目录中拷贝一个新文件，这样目标目录中就比源目录多出一个文件。

```
[root@xuexi ~]# rsync -r /etc/cron.d /tmp/
[root@xuexi ~]# cp /etc/fstab /tmp/cron.d/
[root@xuexi ~]# ls /tmp/cron.d/
0hourly  fstab  raid-check  sysstat
```

再使用"--delete"选项，这时会将目标端多出的文件给删除掉，然后进行同步。

```
[root@xuexi ~]# rsync -r -v /etc/cron.d /tmp --delete
sending incremental file list
deleting cron.d/fstab
cron.d/0hourly
cron.d/raid-check
cron.d/sysstat

sent 704 bytes received 70 bytes 1548.00 bytes/sec
total size is 471 speedup is 0.61
```

这样的行为实现了远程删除的功能，对于作用于本地的rsync，也就实现了rm的本地删除功能。而且，如果使用空目录作为源目录，则它的作用是清空目录上的整个目录。

如果将"--delete"选项和"--exclude"选项一起使用，则被排除的文件不会被删除。例如：

```
[root@xuexi ~]# rsync -r /var/log/anaconda /var/log/audit /tmp  # 先进行一次同步以便测试
[root@xuexi ~]# cp /etc/fstab /tmp/anaconda/                                # 拷贝一个新文件到目标目录以便测试

[root@xuexi ~]# rsync -r -v --exclude="anaconda/*.log" /var/log/anaconda /var/log/audit
/tmp --delete
sending incremental file list
deleting anaconda/fstab
anaconda/syslog
audit/audit.log

sent 3406190 bytes received 52 bytes 6812484.00 bytes/sec
total size is 3405579 speedup is 1.00
```

结果发现只删除了"anaconda/fstab"文件，被"--exclude"规则匹配的anaconda/\*.log文件都没有被删除。也就是网上所说的言论：exclude排除的文件不会被删除。

结论是没错的，但我想很多人不知道为何会如此，也可能从来没想过为何会如此，所以我简单地做个说明。

**在发送端将文件列表发送给接收端后，接收端的generator(要是不知道，你认为是某个就好了)进程会扫描每个文件列表中的信息，然后对列表中的每个信息条目都计算数据块校验码，最后将数据库校验码发给发送端，发送端通过校验码来匹配哪些数据块是需要传输的，这样就实现了增量传输的功能——只传输改变的部分，不会传输整个文件。而delete删除的时间点是generator进程处理每个文件列表时、生成校验码之前进行的，先将目标上存在但源上不存在的多余文件删除，这样就无需为多余的文件生成校验码。**

所以，delete动作是比"--exclude"规则更晚执行的，被"--exclude"规则排除的文件不会进入文件列表中，在执行了delete时会认为该文件不存在于源端，从而导致目标端将这些文件删除。但这是想当然的，尽管理论上确实是这样的，但是rsync为了防止众多误删除情况，提供了两种规则：保护规则(protect)和取消保护规则(risk)。默认情况下，"--delete"和"--exclude"一起使用时，虽然发送端的exclude规则将文件标记为隐藏，使得接收端认为这些被排除文件在源端不存在，但rsync会将这些隐藏文件标记为保护文件，使得它们不受delete行为的影响，这样delete就删除不了这些被排除的文件。如果还是想要强行删除被exclude排除的文件，可以使用"--delete-excluded"选项强制取消保护，这样即使被排除的文件也会被删除。

那么现在，是否理解了网上的言论"exclude排除的文件不会被删除"？

除了"--delete"，相关的选项还有"--delete-before"、"--delete-during"、"--delete-delay"等，它们都隐含了"--delete"选项，它们分别表示generator处理各个文件列表之前一次性全部删除待删除文件、处理文件列表时处理到哪个文件列表就删除该文件列表中的待删除文件，以及同步完所有数据后一次性删除所有待删除文件。

举个例子，假如源端要传输3个目录a、b、c，在目标端a目录中有a1、a2、a3共3个文件需要被删除，b目录中有b1、b2、b3需要删除，同理c目录也一样c1、c2、c3需要被删除。

如果是"--delete-before"，则在目标端rsync刚启动时，就会把a1-a3、b1-b3、c1-c3一次性删除，然后才会处理文件列表中的a目录，处理完a后处理b，再是c。

如果是"--delete-during"，则在目标端rsync刚启动时，先处理文件列表中的a目录，处理a目录时发现此目录中有待删除文件a1-a3，顺手就删除它们，然后完成a目录的相关操作，再处理文件列表中的b目录，发现也有待删除文件b1-b3，顺手删除它们，同理c1-c3也如此。

如果是"--delete-delay"，则同步完文件列表中的a/b/c目录后，最后一次性删除a1-a3、b1-b3、c1-c3。

其实"--delete"选项大多数情况下默认采用的就是"--delete-during"。

## 2.5 rsync daemon模式

### 2.5.1 简介

既然rsync通过远程shell就能实现两端主机上的文件同步，还要使用rsync的服务干啥？试想下，你有的机器上有一堆文件需要时不时地同步到众多机器上去，比如目录a、b、c是专门传输到web服务器上的，d/e、f、g/h是专门传输到ftp服务器上的，还要对这些目录中的某些文件进行排除，如果通过远程shell连接方式，无论是使用排除规则还是包含规则，甚至一条一条rsync命令地传输，这都没问题，但太过繁琐且每次都要输入同样的命令显得太死板。使用rsync daemon就可以解决这种死板问题。而且，rsync daemon是向外提供服务的，这样只要告诉了别人rsync的url路径，外人就能向ftp服务器一样获取文件列表并进行选择性地下载，所以，你所制定的列表，你的同事也可以获取到并使用。

举个简单的例子，Linux内核官网www.kernel.org提供rsync的下载方式，官方给出的地址是rsync://rsync.kernel.org/pub，可以根据这个地址找出你想下载的内核版本。例如要找出linux-3.0.15版本的内核相关文件。

```
[root@xuexi ~]# rsync --no-motd -r -v -f "+ */" -f "+ linux-3.0.15*" -f "- *" -m
rsync://rsync.kernel.org/pub/
receiving file list ... done
drwxr-xr-x    124 2017/07/14 20:27:22 .
drwxr-xr-x    178 2014/11/12 05:50:10 linux
drwxr-xr-x   4096 2017/06/27 05:46:27 linux/kernel
drwxr-xr-x   237568 2017/07/05 20:49:33 linux/kernel/v3.x
-rw-r--r--  76803806 2012/01/04 03:00:31 linux/kernel/v3.x/linux-3.0.15.tar.bz2
-rw-r--r--  96726195 2012/01/04 03:00:31 linux/kernel/v3.x/linux-3.0.15.tar.gz
-rw-r--r--     836 2012/01/04 03:00:31 linux/kernel/v3.x/linux-3.0.15.tar.sign
-rw-r--r--  63812604 2012/01/04 03:00:31 linux/kernel/v3.x/linux-3.0.15.tar.xz

sent 59 bytes  received 80.19K bytes  12.35K bytes/sec
total size is 237.34M  speedup is 2957.66
```

你无需关注上面的规则代表什么意思，需要关注的重点是通过rsync可以向外提供文件列表并提供相应的下载。

同样，你还可以根据路径，将rsync daemon上的文件拉取到本地实现下载的功能。

```
[root@xuexi ~]# rsync --no-motd -avzP
rsync://rsync.kernel.org/pub/linux/kernel/v3.x/linux-3.0.15.tar.bz2 /tmp
receiving incremental file list
linux-3.0.15.tar.bz2
      2834426   3%   300.51kB/s   0:40:22
```

下面就来介绍下rsync daemon。

rsync daemon是"rsync --daemon"或再加上其他一些选项启动的，它会读取配置文件，默认是/etc/rsyncd.conf，并默认监听在873端口上，当外界有客户端对此端口发起连接请求，通过这个网络套接字就可以完成连接，以后与该客户端通信的所有数据都通过该网络套接字传输。

rsync daemon的通信方式和传输通道与远程shell不同。**远程shell连接的两端是通过管道完成通信和数据传输的，即使连接的一端是远程主机，当连接到目标端时，将在目标端上根据远程shell进程fork出rsync进程使其成为rsync server。而rsync daemon是事先在server端上运行好的rsync后台进程（根据启动选项，也可以设置为非后台进程），它监听套接字等待client端的连接，连接建立后所有通信方式都是通过套接字完成的。**

注意，rsync中的server的概念从来就不代表是rsync daemon，server在rsync中只是一种通用称呼，只要不是发起rsync请求的client端，就是server端，你可以认为rsync daemon是一种特殊的server，其实daemon更应该称之为service。（之所以解释这一点，是避免各位初学的朋友在阅读man rsync过程中产生误解）

以下是rsync client连接rsync daemon时的命令语法：

```
Pull: rsync [OPTION...] [USER@]HOST::SRC... [DEST]
      rsync [OPTION...] rsync://[USER@]HOST[:PORT]/SRC... [DEST]
Push: rsync [OPTION...] SRC... [USER@]HOST::DEST
      rsync [OPTION...] SRC... rsync://[USER@]HOST[:PORT]/DEST
```

连接命令有两种类型，一种是rsync风格使用双冒号的"rsync user@host::src dest"，一种是url风格的"rsync://user@host:port/src dest"。对于rsync风格的连接命令，如果想要指定daemon号，则需要使用选项"--port"。

上述语法中，其中daemon端的路径，如user@host::src，它的src代表的是模块名，而不是真的文件系统中的路径。关于rsync中的模块，相信见了下面的配置文件就会知道是什么意思。

## 2.5.2 daemon配置文件rsyncd.conf

默认"rsync --daemon"读取的配置文件为/etc/rsyncd.conf，有些版本的系统上可能该文件默认不存在。rsyncd.conf的配置见man rsyncd.conf。以下是部分内容：

```
[root@xuexi ~]# cat /etc/rsyncd.conf
# /etc/rsyncd: configuration file for rsync daemon mode

# See rsyncd.conf man page for more options.

# configuration example:

# uid = nobody
# gid = nobody
# use chroot = yes
# max connections = 4
# pid file = /var/run/rsyncd.pid
# exclude = lost+found/
# transfer logging = yes
# timeout = 900
# ignore nonreadable = yes
# dont compress = *.gz *.tgz *.zip *.z *.Z *.rpm *.deb *.bz2

# [ftp1]
#       path = /home/ftp
#       comment = ftp export area
```

在上述示例配置文件中，先定义了一些全局选项，然后定义了[ftp1]，这个用中括号包围的"[ftp1]"就是rsync中所谓的模块，ftp1为模块ID，必须保证唯一，每个模块中必须定义一项"path"，path定义的是该模块代表的路径，例如此示例文件中，如果想请求ftp1模块，则在客户端使用"rsync user@host::/ftp1"，这表示访问user@host上的/home/ftp目录，如果要访问/home/ftp目录下的子目录www，则"rsync user@host::/ftp1/www"。

以下是常见的配置项，也算是一个配置示例：

```
##### 全局配置参数 #####
port=888      # 指定rsync端口。默认873
uid = rsync # rsync服务的运行用户，默认是nobody，文件传输成功后属主将是这个uid
gid = rsync # rsync服务的运行组，默认是nobody，文件传输成功后属组将是这个gid
use chroot = no # rsync daemon在传输前是否切换到指定的path目录下，并将其监禁在内
max connections = 200 # 指定最大连接数量，0表示没有限制
timeout = 300        # 确保rsync服务器不会永远等待一个崩溃的客户端，0表示永远等待
motd file = /var/rsyncd/rsync.motd # 客户端连接过来显示的消息
pid file = /var/run/rsyncd.pid      # 指定rsync daemon的pid文件
lock file = /var/run/rsync.lock     # 指定锁文件
log file = /var/log/rsyncd.log      # 指定rsync的日志文件，而不把日志发送给syslog
dont compress = *.gz *.tgz *.zip *.z *.Z *.rpm *.deb *.bz2 # 指定哪些文件不用进行压缩传输

#####下面指定模块，并设定模块配置参数，可以创建多个模块#####
[longshuai]      # 模块ID
path = /longshuai/ # 指定该模块的路径，该参数必须指定。启动rsync服务前该目录必须存在。rsync请求访问模块本质就是访问该路径。
ignore errors      # 忽略某些IO错误信息
read only = false   # 指定该模块是否可读写，即能否上传文件，false表示可读写，true表示可读不可写。所有模块默认不可上传
write only = false # 指定该模式是否支持下载，设置为true表示客户端不能下载。所有模块默认可下载
list = false        # 客户端请求显示模块列表时，该模块是否显示出来，设置为false则该模块为隐藏模块。默认true
hosts allow = 10.0.0.0/24 # 指定允许连接到该模块的机器，多个ip用空格隔开或者设置区间
hosts deny = 0.0.0.0/32    # 指定不允许连接到该模块的机器
auth users = rsync_backup # 指定连接到该模块的用户列表，只有列表里的用户才能连接到模块，用户名和对应密码保存在secrets file中，
                           # 这里使用的不是系统用户，而是虚拟用户。不设置时，默认所有用户都能连接，但使用的是匿名连接
secrets file = /etc/rsyncd.passwd # 保存auth users用户列表的用户名和密码，每行包含一个username:passwd。由于"strict modes"
                                   # 默认为true，所以此文件要求非rsync daemon用户不可读写。只有启用了auth users该选项才有效。
[xiaofang]      # 以下定义的是第二个模块
path=/xiaofang/
read only = false
ignore errors
comment = anyone can access
```

注意：

(1).客户端推到服务端时，文件的属主和属组是配置文件中指定的uid和gid。但是客户端从服务端拉的时候，文件的属主和属组是客户端正在操作rsync的用户身份，因为执行rsync程序的用户为当前用户。

(2).auth users和secrets file这两行不是一定需要的，省略它们时将默认使用匿名连接。但是如果使用了它们，则secrets file的权限必须是600。客户端的密码文件也必须是600。

(3).关于secrets file的权限，实际上并非一定是600，只要满足除了运行rsync daemon的用户可读即可。是否检查权限的设定是通过选项strict mode设置的，如果设置为false，则无需关注文件的权限。但默认是yes，即需要设置权限。

配置完后，再就是提供模块相关目录、身份验证文件等。

```
[root@xuexi ~]# useradd -r -s /sbin/nologin rsync
[root@xuexi ~]# mkdir /{longshuai,xiaofang}
[root@xuexi ~]# chmod -R rsync.rsync /{longshuai,xiaofang}
```

提供模块longshuai身份验证文件，由于rsync daemon是以root身份运行的，所以要求身份验证文件对非root用户不可读写，所以设置为600权限。

```
[root@xuexi ~]# echo "rsync_backup:123456" >> /etc/rsyncd.passwd
[root@xuexi ~]# chmod 600 /etc/rsyncd.passwd
```

然后启动rsync daemon，启动方式很简单。

```
[root@xuexi ~]# rsync --daemon
```

如果是CentOS 7，则自带启动脚本。

```
[root@xuexi ~]# systemctl start rsyncd
```

看看该脚本的内容。

```
[root@xuexi ~]# cat /usr/lib/systemd/system/rsyncd.service
[Unit]
Description=fast remote file copy program daemon
ConditionPathExists=/etc/rsyncd.conf

[Service]
EnvironmentFile=/etc/sysconfig/rsyncd
ExecStart=/usr/bin/rsync --daemon --no-detach "$OPTIONS"

[Install]
WantedBy=multi-user.target
```

可以看到启动方法也仅仅只是多了一个"--no-detach"，该选项表示rsync不将自己从终端上剥离。

总之，启动好rsync daemon后，它就监听在指定的端口上，等待客户端的连接。

由于上述示例中的模块longshuai配置了身份验证功能，所以客户端连接时会询问密码。如果不手动输入密码，则可以使用"--password-file"选项提供密码文件，密码文件中只有第一行才是传递的密码，其余所有的行都会被自动忽略。

例如在客户端上：

```
[root@xuexi ~]# echo "123456" > /tmp/rsync_passwd
```

然后使用该"--password-file"连接需要身份验证的longshuai模块。

```
[root@xuexi ~]# echo "123456" > /tmp/rsync_passwd
```

如果需要访问模块中的某个文件，则：

```
[root@xuexi ~]# rsync --list-only --port 888 rsync_backup@172.16.10.6::longshuai/a/b --
password-file=/tmp/rsync_passwd
```

还可以使用url格式语法：

```
[root@xuexi ~]# rsync --list-only rsync://rsync_backup@172.16.10.6:888/longshuai/a/b --
password-file=/tmp/rsync_passwd
```

## 2.6 远程shell方式连接使用daemon

在前文说了rsync有三种工作方式：本地同步模式、远程shell模式和rsync daemon模式。前两者是使用管道进行通信和传输数据的，后者是通过网络套接字进行通信和传输数据的，且rsync daemon要求在server端必须已经运行好rsync且监听在指定端口上。

但rsync支持第4种工作方式：通过远程shell方式连接rsync daemon。也就是将第二种和第三种方式结合起来。虽然这种方式用的不多，但还是有必要稍微解释下，为你阅读rsync的man文档提供一些帮助。

为了下面称呼的方便，暂且将通过远程shell连接使用daemon的方式成为"远程shell daemon"，当然，官方并没有这样的术语，仅仅只是本人在此为了方便而如此称呼。

远程shell daemon的方式严格地说是"远程shell通信方式+使用rsync daemon的功能"。所以它的通信方式和远程shell是一样的，在客户端发起远程shell连接，在server端fork远程shell进程以启动rsync进程，但这个rsync进程是临时的rsync daemon，它只读取配置文件中client所请求的模块部分，且只读取模块部分中的path和身份认证相关内容，（也就是说不会将全局配置项和其它模块项加载到内存，该模块下的其他配置也不会生效），当rsync操作完成，该rsync daemon就消逝并从内存中被清理。而且，远程shell daemon启动的临时daemon不会和已经在server端运行的rsync daemon冲突，它们可以并存。由于远程shell连接的最终目标是rsync模块，所以它只能使用rsync daemon语法。

以下是语法格式：为了简洁，没有指定src还是dest，且以ssh这个远程shell为例。

```
rsync [options] --rsh=ssh auth_user@host::module
rsync [options] --rsh="ssh -l ssh_user" auth_user@host::module
rsync [options] -e "ssh -l ssh_user" auth_user@host::module
rsync [options] -e "ssh -l ssh_user" rsync://auth_user@host/module
```

涉及了两个用户ssh\_user和auth\_user，由于使用的是远程shell通信方式，所以client要和server端建立ssh连接，ssh\_user就是ssh连接server的用户。auth\_user则是模块中的身份认证用户。如果不指定"ssh\_user"，则默认将使用auth\_user，但很多时候auth\_user都只是一个虚拟用户，这样就建立了ssh连接导致失败，所以建议明确指定ssh\_user和auth\_user。

举个例子就能说明上面的一切。以下是server端配置文件/etc/rsyncd.conf中的一个模块配置，稍后将从client端使用远程shell方式请求该模块。

```
[tmpdir]
path=/tmp
auth users=lisi
secrets file=/tmp/lisi_passwd
```

当前server端是没有rsync daemon在运行的。

```
[root@xuexi ~]# netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State
tcp      0      0 0.0.0.0:22              0.0.0.0:*              LISTEN
tcp      0      0 127.0.0.1:25             0.0.0.0:*              LISTEN
tcp6     0      0 ::1:22                  ::*:*                 LISTEN
tcp6     0      0 ::1:25                  ::*:*                 LISTEN
```

在客户端上使用以下命令：

```
[root@xuexi ~]# rsync --list-only -e "ssh -l root" lisi@172.16.10.6::tmpdir
root@172.16.10.6's password:
Password:
```

可以看到要求输入两次密码，第一次密码是root@XXX的密码，即建立ssh连接使用的密码，只有建立了ssh连接，才能在server上启动临时rsync daemon。第二次输入的密码Password是"auth users=lisi"对应的密码。

回到系列文章大纲：<http://www.cnblogs.com/f-ck-need-u/p/7048359.html>

**转载请注明出处：**

**<http://www.cnblogs.com/f-ck-need-u/p/7220009.html>**

分类: Linux服务篇



