日志服务和文件共享

讲师：王树森

1 日志服务和文件共享

1.1 rsyslog

1.1.1 日志服务

1.1.1.1 日志是干什么的

基础知识

简介

```
    Linux日志服务管理主要涉及对系统中产生的各种日志文件进行收集、分析、备份、轮转和删除等操作，以便监控系统的运行状况、诊断和解决问题，并提高系统的安全性和性能。
    日志记录的内容包括：
    - 历史事件：时间，地点，人物，事件
    - 日志级别：事件的关键性程度，Loglevel
```

场景需求

```
    在现实生活中，记录日志非常重要，比如：银行转账时会有转账记录；飞机飞行过程中的黑盒子（飞行数据记录器）记录着飞机的飞行过程，那么将系统和应用发生的事件记录至日志中，也很意义，常可以助于排错和分析使用。
```

某名人日记

```
7月4日    新开这本日记，也为了督促自己下个学期多下些苦功。先要读完手边的莎士比亚的《亨利八世》。
7月13日   打牌。
7月14日   打牌。
7月15日   打牌。
...
10月23日  打牌。不行，xx你不能再这么堕落下去了，明天开始就要学习。
10月23日  打牌。
...
```

某国王日记

```
7月14日
    今日无事
次日，国家暴动，国王被杀
```

某网文日记

```
    “10月20日，天气阴，我先不急着辞职，把生意搞起来再辞职。...今日在衙门捡到一钱银子。”
    “10月21日，天气晴，...，我在勾栏捡到了一钱银子，正好用来支付听曲吃菜的钱....最近是不是走了狗屎运？”
    “10月22日，勾栏听曲。”
    “10月23日，勾栏听曲。”
    “10月24日，勾栏听曲，王捕头问我为何如此快乐？因为白嫖使我快乐。”
    “10月25日，...，你怎可如此堕落，不能这样下去了，你忘记自己的目标了吗？先订个小目标，赚一个亿。”
    “10月26日，勾栏听曲。”
    “10月27日，勾栏听曲。今天没有捡到银子，我支付了一钱的piao资。呸，乌烟瘴气的地方，再也不来了。”
    “10月28日，这个世界...对了，玻璃！我可以烧玻璃，玻璃可是好东西啊，这群古代人肯定没见过。”
    “10月29日，哦，玻璃也有了，我得另谋出路。今日在家里捡到二叔的私房钱，一钱银子。”
    “10月30日，勾栏听曲。”
```

1.1.1.2 常见的系统日志

sysklogd 系统日志服务

```
    在 CentOS5 以及之前的发行版中，其采用的 sysklogd 服务来记录和管理系统日志的。sysklogd 服务有两个模块：
- klogd： 用于记录 linux kernel 相关的日志
- syslogd：用于记录用户空间应用日志
```

rsyslog 系统日志服务

```
    RSYSLOG 就是 Rocket-fast SYStem for LOG processing，即极速日志处理系统。rsyslog 是 CentOS6 以后的版本中使用的日志管理程序，是一个默认安装的服务，并且默认开机启动。
    官方网站：https://www.rsyslog.com/
```

```
Rocky系统
    [root@rocky9 ~]# rpm -q rsyslog
    rsyslog-8.2310.0-4.el9.x86_64

Ubuntu系统
    root@ubuntu24:~# dpkg -l rsyslog

openEuler系统
    [root@sswang ~]# rpm -q rsyslog
    rsyslog-8.2312.0-3.oe2403.x86_64
```

```
root@ubuntu24:~# dpkg -l rsyslog
期望状态=未知(u)/安装(i)/删除(r)/清除(p)/保持(h)
| 状态=未安装(n)/已安装(i)/仅存配置(c)/仅解压缩(U)/配置失败(F)/不完全安装(H)/触发器等待(W)/触发器未决(T)
|/ 错误?=(无)/须重装(R) (状态，错误：大写=故障)
||/ 名称           版本              体系结构     描述
+++-==============-=================-============-=======================================
ii  rsyslog        8.2312.0-3ubuntu9 amd64        reliable system and kernel logging daemon
root@ubuntu24:~#
```

ELK 日志服务

```
    ELK 是三个开源软件的缩写，分别表示：Elasticsearch , Logstash, Kibana , 它们都是开源软件。在后续发展的过程中增加了一个 FileBeat，这几款软件通常在一起配合使用，各司其职。ELK 主要用于部署在企业架构中，收集多台设备上多个服务的日志信息，并将其统一整合后提供给用户。
```

rsyslog vs ELK

```
rsyslog 主要用于单机日志管理，ELK 主要用于分布式集群环境中的日志管理。
```

1.1.1.3 rsyslog基础

基础知识

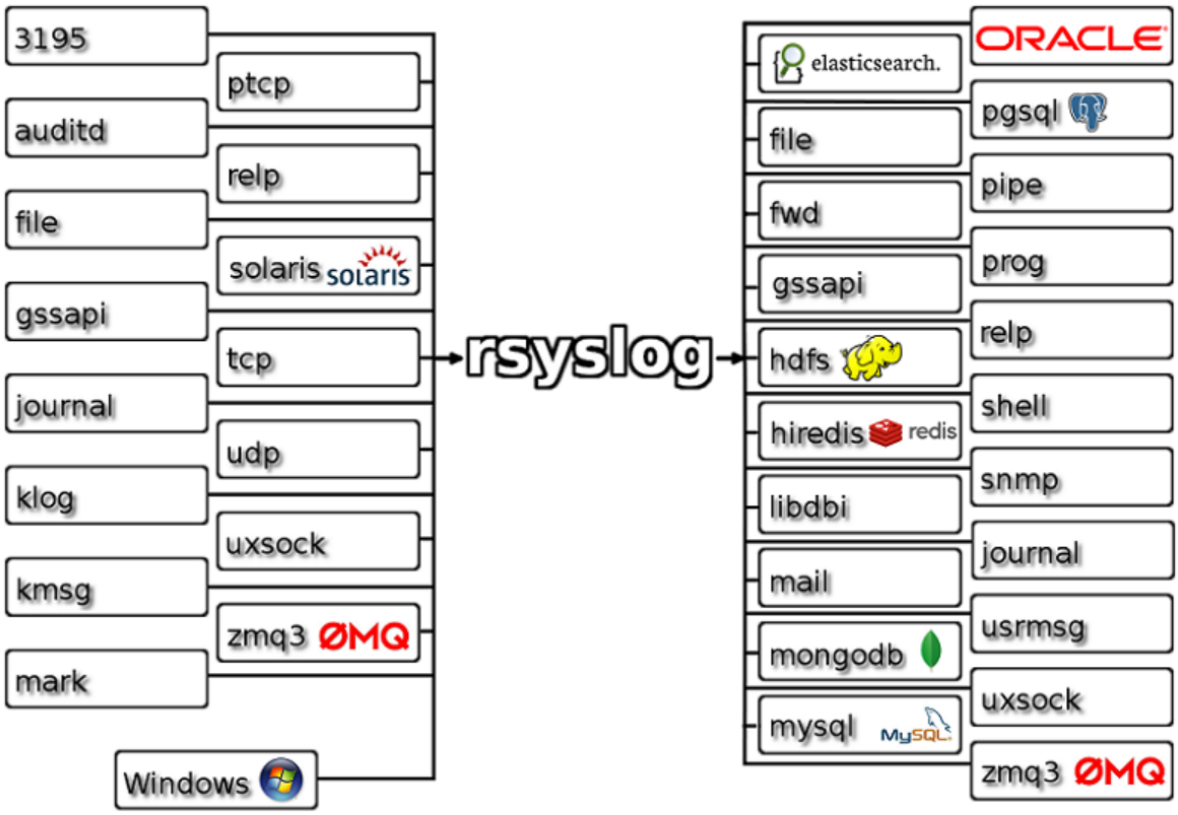
简介

```
    Rsyslog是一个开源工具，基于syslog协议完成系统日志的处理转发，它可用于接受来自各种来源的输入，转换它们，并将结果输出到不同的目的地。
    Rsyslog提供了高性能、强大的安全功能和模块化设计，支持多种输入和输出模块，可以与各种设备和系统集成，例如syslog、TCP、UDP、TLS等。同时，它还支持多种过滤和处理规则，可以根据需求定制化日志的存储、转发和处理方式。
```

```
查看帮助：https://www.rsyslog.com/doc/index.html
```

rsyslog 特性

```
1. 支持输出日志到各种数据库，如 MySQL，PostgreSQL，MongoDB ElasticSearch，实现使用第三方服务对日志进行存储和分析；
2. 精细的输出格式控制以及对日志内容的强大过滤能力，可实现过滤记录日志信息中的指定部份；
3. 通过 RELP + TCP 实现数据的可靠传输
4. 支持数据的加密和压缩传输等
5. 多线程
```



信息查看

查看rsyslog基本信息

```
查看所有 rsyslog 包中的文件
root@ubuntu24:~# dpkg -L rsyslog
[root@rocky9 ~]# rpm -ql rsyslog
/usr/sbin/rsyslogd                                  #主程序
/usr/lib/systemd/system/rsyslog.service             #服务脚本
/etc/rsyslog.conf                                   #主配置文件
/etc/rsyslog.d/*.conf                               #配置文件目录中的配置文件
/usr/lib/rsyslog/*.so                               #库文件
```

```
rsyslog服务默认开机自启动
[root@openEuler ~]# systemctl is-active rsyslog
active
root@ubuntu24:~# systemctl is-active rsyslog
active
[root@rocky9 ~]# systemctl is-active rsyslog
active
```

查看syslogd程序打开的文件

```
root@ubuntu24:~# lsof -c rsyslogd
COMMAND   PID    USER    FD    TYPE             DEVICE SIZE/OFF       NODE NAME
...
rsyslogd 1126 root    4u     unix 0xffff8ddc038a5400      0t0      29529
type=DGRAM (CONNECTED)
rsyslogd 1126 root    5w      REG               253,0  3250296  68107754
/var/log/messages
...
```

通过id方式查看

```
root@ubuntu24:~# lsof -p 1126
```

```
[root@rocky9 ~]# lsof -c rsyslogd
COMMAND   PID USER   FD      TYPE             DEVICE SIZE/OFF      NODE NAME
rsyslogd 1126 root  cwd       DIR              253,0     4096       128 /
rsyslogd 1126 root  rtd       DIR              253,0     4096       128 /
rsyslogd 1126 root  txt       REG              253,0   800312   1728729 /usr/sbin/rsyslogd
rsyslogd 1126 root  mem       REG               0,24  4571136        67 /run/log/journal/aa03379149d74770bc423657e9b6b442/system@1e6b8494e06747f189b2e5d4070fd15f-0000000000000001-000624b5d3268c70.journal
rsyslogd 1126 root  mem       REG               0,24  4571136      1771 /run/log/journal/aa03379149d74770bc423657e9b6b442/system.journal
rsyslogd 1126 root  mem       REG              253,0    37784 202617319 /usr/lib64/rsyslog/imjournal.so
rsyslogd 1126 root  mem       REG              253,0    42320 202617326 /usr/lib64/rsyslog/imuxsock.so
rsyslogd 1126 root  mem       REG              253,0   153600    435294 /usr/lib64/libgpg-error.so.0.32.0
rsyslogd 1126 root  mem       REG              253,0   144144    481778 /usr/lib64/liblz4.so.1.9.3
rsyslogd 1126 root  mem       REG              253,0   882384    420796 /usr/lib64/libzstd.so.1.5.1
rsyslogd 1126 root  mem       REG              253,0   178744    420790 /usr/lib64/liblzma.so.5.2.5
rsyslogd 1126 root  mem       REG              253,0  1288480    481510 /usr/lib64/libgcrypt.so.20.4.0
rsyslogd 1126 root  mem       REG              253,0    36304    435139 /usr/lib64/libcap.so.2.48
rsyslogd 1126 root  mem       REG              253,0  2592552    420611 /usr/lib64/libc.so.6
rsyslogd 1126 root  mem       REG              253,0   108136       145 /usr/lib64/libgcc_s-11-20231218.so.1
rsyslogd 1126 root  mem       REG              253,0    32528    435197 /usr/lib64/libcap-ng.so.0.0.0
rsyslogd 1126 root  mem       REG              253,0    36440    435155 /usr/lib64/libuuid.so.1.3.0
rsyslogd 1126 root  mem       REG              253,0   915856   1814106 /usr/lib64/libsystemd.so.0.35.0
rsyslogd 1126 root  mem       REG              253,0    52856   1121534 /usr/lib64/libfastjson.so.4.3.0
rsyslogd 1126 root  mem       REG              253,0    16032   1121495 /usr/lib64/libestr.so.0.0.0
rsyslogd 1126 root  mem       REG              253,0   914376    420615 /usr/lib64/libm.so.6
rsyslogd 1126 root  mem       REG              253,0   102552    420773 /usr/lib64/libz.so.1.2.11
rsyslogd 1126 root  mem       REG              253,0    32904 202617327 /usr/lib64/rsyslog/lmnet.so
rsyslogd 1126 root  mem       REG              253,0   861832    420479 /usr/lib64/ld-linux-x86-64.so.2
```

1.1.1.4 系统日志相关概念

facility /fəˈsɪləti/ : 设施，从功能或程序上对日志进行归类

在一台主机上会同时运行多个服务和软件，每个服务或软件都有可能会产生大量的日志，如果每个服务或软件产生的日志都独立存放管理，那文件数量就太多了，如果都放到一个文件中，似乎也不是很合适，所以 rsyslog 将日志进行了分类，相同类型的日志放一个文件，这样便于管理。

```
syslog 内置分类
    LOG_AUTH                    #auth 安全和认证相关的日志
    LOG_AUTHPRIV                #authpriv 安全和认证相关的日志，私有
    LOG_CRON                    #cron 系统定时任务 crontab 与 at 产生的相关日志
    LOG_DAEMON                  #daemon 各守护进程产生的日志
    LOG_FTP                     #ftp ftp守护进程产生的日志
    LOG_KERN                    #kern 内核产生的日志
    LOG_LOCAL0 -- LOG_LOCAL7    #local0-local7 自定义分类
    LOG_LPR                     #lpr 打印服务日志
    LOG_MAIL                    #mail 邮件服务日志
    LOG_NEWS                    #news 网络新闻服务产生的日志
    LOG_SYSLOG                  #syslog syslogd 服务自己的日志
    LOG_USER                    #user 用户等级
    LOG_UUCP                    #uucp uucp子系统的日志信息
    *                           #通配符，代表所有分类
```

在 rsyslog 中，LOG_LOCAL0 到 LOG_LOCAL7 是一组特殊的日志设施（facility），它们被设计为供本地使用或开发者自定义的软件类别使用。这些设施允许系统管理员或开发者将特定服务的日志消息定向到指定的日志文件或执行特定的动作。

具体来说，每个 LOG_LOCAL 设施（从 LOG_LOCAL0 到 LOG_LOCAL7）都可以被视为一个自定义的日志通道，用于收集和记录来自特定应用程序或服务的日志信息。

注意：在rsyslog中，LOG_LOCAL0 到 LOG_LOCAL7 并不代表日志的级别，它仅用于自定义日志分类的设施标识。

Priority /praɪˈɔːrəti/: 优先级别，从高到低排序

rsyslog 在记录日志的时候，将各种产生日志的事件和行为进行了优先级的排序，使用者可以根据不同环境(测试/生产)和需求，设置不同的级别来记录日志，这样可以保证，记录下来的内容都是是我们想要的。

rsyslog 内置优先级分类，从高到低，如果在记录日志时，设置了优先级，则只会记录设定的优先级和高于设定优先级的日志

```
LOG_EMERG                   #emerg/panic 紧急，致命错误
LOG_ALERT                   #alert 告警，当前状态必须立即进行纠正
LOG_CRIT                    #crit 关键状态的警告，例如 硬件故障
LOG_ERR                     #err/error 其它错误
LOG_WARNING                 #warning/warn 警告级别的信息
LOG_NOTICE                  #notice 通知级别的信息
LOG_INFO                    #info 通告级别的信息
LOG_DEBUG                   #debug 调试程序时的信息
*                           #所有级别的日志
none                        #不需要任何日志
                            #panic,error,warn在新版中被弃，不建议使用
```

上面的内容，可以通过

```
man 3 syslog 来进行查看
```

1.1.2 配置解读

1.1.2.1 配置结构

基础知识

配置文件

```
查看配置文件
[root@rocky9 ~]# ls /etc/rsyslog.conf
/etc/rsyslog.conf

Rocky9 和 openEuler系统下子配置文件为空
[root@rocky9 ~]# ls /etc/rsyslog.d/
[root@rocky9 ~]#
[root@openEuler ~]# ls /etc/rsyslog.d/
[root@openEuler ~]#

Ubuntu系统下子配置文件有内容
root@ubuntu24:~# ls /etc/rsyslog.d/
20-ufw.conf  21-cloudinit.conf  50-default.conf
```

在主配置文件中，将配置分为三个部份

```
[root@rocky9 ~]# grep '#### ' /etc/rsyslog.conf
#### GLOBAL DIRECTIVES ####                    # 全局配置(GLOBAL DIRECTIVES)
#### MODULES ####                              # 模块(MODULES)
#### RULES ####                                # 日志记录规则配置(RULES)
注意：
    对于Ubuntu系统来说，它的Rules内容在子配置文件里面
```

```
全局配置(GLOBAL DIRECTIVES)内容示例：
# Where to place auxiliary files
global(workDirectory="/var/lib/rsyslog")
# Use default timestamp format
module(load="builtin:omfile" Template="RSYSLOG_TraditionalFileFormat")

Ubuntu系统中的内容示例
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat    #默认日志模板
$RepeatedMsgReduction on                                    #默认开启重复过滤
$FileOwner syslog                                           #创建日志文件的默认权限和
属主属组
$FileGroup adm
$FileCreateMode 0640
$DirCreateMode 0755
$Umask 0022
$PrivDropToUser syslog
$PrivDropToGroup syslog
$WorkDirectory /var/spool/rsyslog                           #工作目录，默认目录为空
$IncludeConfig /etc/rsyslog.d/*.conf                        #独立配置文件引用目录
```

```
模块(MODULES)内容示例：
# Provides UDP syslog reception
# for parameters see http://www.rsyslog.com/doc/imudp.html
#module(load="imudp") # needs to be done just once
#input(type="imudp" port="514")          # UDP模块，默认没有启用

# Provides TCP syslog reception
# for parameters see http://www.rsyslog.com/doc/imtcp.html
#module(load="imtcp") # needs to be done just once
#input(type="imtcp" port="514"           # TCP模块，默认没有启用
```

```
日志记录规则配置(RULES)内容示例:
authpriv.*                                      /var/log/secure
mail.*                                          -/var/log/maillog
cron.*                                          /var/log/cron
*.emerg                                         :omusrmsg:*
uucp,news.crit                                  /var/log/spooler
local7.*                                        /var/log/boot.log
```

如何查看帮助

```
[root@rocky9 ~]# yum install rsyslog-doc -y
root@ubuntu24:~# apt install rsyslog-doc -y

查看帮助
man rsyslogd
```

1.1.2.2 配置规则

rule简介

```
    每一行 rule 由两列组成，分别是选择器(selector)和处理动作(action)，选择器将过滤后的日志交由处理动作处理;
    选择器可以同时有多个，用分号分隔，处理动作也可以同时有多个，用 & 分隔，处理动作中可以指定模板，不同的模板会生成不同的日志内容，模板可以自定义。
```

```
格式:
    facility.priority[;facility.priority;...]       action
```

配置样例解读

```
# kern的所有级别日志都记录到/var/log/secure文件中
kern.*                                                  /dev/console

# 除了mail、authpriv、cron 之外的所有info级别日志记录到 /var/log/messages文件里面
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# authpriv的所有级别日志都记录到/var/log/secure文件中
authpriv.*                                              /var/log/secure
mail.*                                                  -/var/log/maillog
cron.*                                                  /var/log/cron

# 紧急致命级别的错误信息，调用 omusrmsg 模块发给所有登录用户
*.emerg                                                 :omusrmsg:*
uucp,news.crit                                          /var/log/spooler
local7.*                                                /var/log/boot.log
```

Rocky 中常见日志说明

```
[root@rocky9 ~]# cat /etc/rsyslog.conf | grep -Ev "^#|^$"

*.info;mail.none;authpriv.none;cron.none    /var/log/messages        #除了
mail,authpriv,cron 之外都记录
authpriv.*                                  /var/log/secure          #安全认证相关日
志
mail.*                                      -/var/log/maillog        #邮件服务相关日
志
cron.*                                      /var/log/cron            #定时任务相关日
志
*.emerg                                     :omusrmsg:*              #所有致命错误信
息，调用omusrmsg发给所有登录用户
uucp,news.crit                              /var/log/spooler         #uucp,新闻相关
日志
local7.*                                    /var/log/boot.log        #操作系统启动流
程日志
```

ubuntu 中常见日志说明

```
[root@ubuntu24 ~]# cat /etc/rsyslog.d/*conf | grep -Ev "^#|^$"
:msg,contains,"[UFW " /var/log/ufw.log                              #ufw 服务日志
:syslogtag, isequal, "[CLOUDINIT]" /var/log/cloud-init.log          #cloud-init
服务日志
& stop                                                              #其它不处理
auth,authpriv.*         /var/log/auth.log
#auth,authpriv 的日志
*.*;auth,authpriv.none      -/var/log/syslog                        #除了
auth,authpriv之外的日志
kern.*              -/var/log/kern.log                              #内核产生的日志
mail.*              -/var/log/mail.log                              #邮件服务日志
mail.err            /var/log/mail.err                               #邮件服务
err(含)以上日志
*.emerg             :omusrmsg:*                                     #所有致命错误信
息，调用omusrmsg发给所有登录用户
```

*.emerg 表示所有设施产生的紧急级别（Emergency）的日志消息。
: 这个符号用于分隔选择器和动作。在 rsyslog 配置中，它告诉 rsyslog，左边的部分（选择器和优先级）指定了哪些日志消息应该被处理，而右边的部分（动作）指定了这些日志消息应该如何被处理。
omusrmsg:*：这部分指定了对符合条件的日志消息执行的动作。omusrmsg 是一个输出模块（Output Module），用于将日志消息发送给指定的用户。在这个上下文中，omusrmsg 后面跟着的 * 表示将日志消息发送给系统上的所有用户。具体来说，它会通过写入到所有用户的终端（通常是他们的登录会话）来显示这些紧急日志消息。

上面规则的目的：
　　将所有设施产生的紧急级别（Emergency）的日志消息发送给系统上的所有用户。这通常用于在发生严重系统错误或紧急情况时，立即通知所有用户。然而，这种配置可能更多地出现在旧系统或需要确保紧急消息能够立即引起所有用户注意的特殊环境中。

1.1.2.3 选择器和动作

选择器的多种写法

```
选择器有以下几种定义方式：
    - 用分类和优先级来过滤，同一条 rule 中，分类和优先级都可以有多个，用逗号分隔
    - 基于日志内容中的指定字段来过滤
    - 基于表达式构建脚本来过滤

facility 优先级的多种写法
    *               所有 priority
    none            没有任何 priority，即不记录
    priority        具体的 priority，处理指定级别和指定级别以上的所有级别日志
    =priority       仅处理指定级别日志
    !priority       排除指定的 priority，这种写法不能单独使用
```

```
selector-基于日志内容指定字段过滤

字段，比较表达式，要比较的值
    :property, [!]compare-operation, "value"  action

所有可用 property
    https://www.rsyslog.com/doc/master/configuration/properties.html

比较表达式
    contains            字段是否包含 value 的内容
    contains_i          字段是否包含 value 的内容，不区分大小写
    isequal             字段的内容是否与 value 相等
    startswith          是否以 value 开头
    startswith_i        是否以 value 开头，不区分大小写
    regex               基本正则
    ereregex            扩展正则
    isempty             是否为空，不需要后面的 value
```

```
selector-基于RainerScript设置更复杂的日志过滤
    https://www.rsyslog.com/doc/v8-stable/rainerscript/index.html
```

动作的写法

```
处理动作有以下几种：
    - 输出到日志文件或某个特定设备
    - 保存到数据库
    - 发送给指定用户，该用户必须已登录，可以同时指定多个用户，用逗号分隔
    - 传送到远程主机
    - 通过管道传送给其它命令
    - 丢弃日志，不处理
```

```
action-输出到日志文件或设备
    /path/file      将日志内容写到指定文件
    -/path/file     将日志内容写到指定文件，异步写入
    /dev/null       将日志内容输出到指定设备
```

```
action-保存到数据库，保存到数据库要开启相应模块
    module (load="ommysql")
    *.* action(type="ommysql" server="10.0.0.210" db="rsyslog" uid="rsysloger"
pwd="123456")
```

```
action-发送给指定用户
    root                    将日志内容发送给用户 root
    root,tom                将日志内容发送给用户 root 和 tom
    *                       将日志内容发送给所有已登录用户
```

```
action-发送到远程主机
    @192.168.2.123                  使用 UDP 发送到远程主机，默认端口514
    @@log.magedu.com:256            使用 TCP 发送到远程主机 256 端口[修改的]，默认端口
514
    @(z6)[fe80::20c:29ff:fe7e:ce82] 使用 UDP 发送到远程主机(IPV6地址)，启用zlib压缩，压
缩级别为6
    注意：
        @：这个前缀表示使用UDP协议来发送日志。
        @@：这个前缀表示使用TCP协议来发送日志。
            提供了更可靠的日志传输，因为它包含错误检查和重传机制，但可能会稍微慢一些。
        (z6)：这部分是rsyslog的框架或协议指示器的一部分，rsyslog支持多种输出模块和协议，
            但(z6)并不是标准配置语法，而是扩展中的语法，用于指定特定行为或参数的标记。
            这里面表示启用zlib压缩，压缩级别为6，
```

```
action-通过管道传送给其它命令，管道必须有名管道，要事先创建，此功能在 rsyslog8 版本后才支持
```

```
action-不处理
    stop                            不处理，丢弃
```

1.1.3 日志模板

基础知识

简介

```
    日志模板是rsyslog中的一个重要概念，它定义了日志消息的格式和内容。通过定义模板，管理员可以控
制日志中显示的信息类型、顺序和格式。这有助于管理员更好地理解日志内容，快速定位问题。
```

模版组成

```
日志模板通常由多个属性组成，这些属性可以从日志消息中提取特定的信息。以下是一些常见的rsyslog属
性：
    timestamp：时间戳，记录日志事件发生的时间。
    hostname：主机名，记录产生日志事件的服务器的主机名。
    syslogtag：syslog标签，通常与产生日志的进程相关联。
    msg：日志消息的具体内容。
此外，还有一些系统属性，如$NOW（当前日期戳）、$YEAR（当前年份）、$MONTH（当前月份）等，这些属
性可以进一步丰富日志模板的内容。
```

在rsyslog中，可以通过四种方式定义日志模板

1 在配置文件中直接定义：

在rsyslog的配置文件（如/etc/rsyslog.conf或/etc/rsyslog.d/目录下的文件）中，可以使用template语句来定义模板。例如：

```
template(name="MyTpl" type="list") {
    property(name="timestamp" dateFormat="rfc3339")
    constant(value=" ")
    property(name="hostname")
    constant(value=" ")
    property(name="syslogtag")
    constant(value=" ")
    property(name="msg" spifno1stsp="on" droplastlf="on")
    constant(value="\n")
}
```

2 使用旧式语法：

rsyslog还支持旧式的模板语法，这种语法在早期的rsyslog版本中广泛使用。例如：

```
$template MyTpl,"%timestamp::date-rfc3389% %HOSTNAME% %syslogtag% %msg%\n"
```

这个旧式语法的模板定义了与上面示例相似的属性，但使用了不同的格式和占位符。

3 隐式方式定制日志模版

日志内容由 template 决定，如果没有显式指定，默认使用 RSYSLOG_TraditionalFileFormat，其具体内容如下：

```
    template(name="RSYSLOG_TraditionalFileFormat" type="string"
     string="%TIMESTAMP% %HOSTNAME% %syslogtag%%msg:::sp-if-no-1st-sp%%msg:::drop-last-lf%\n")
```

rsyslog 中有13个内置的模板，我们可以在配置文件中直接使用，其名称如下，具体定义的内容需要查询相关文档

```
    RSYSLOG_TraditionalFileFormat
    RSYSLOG_FileFormat
    RSYSLOG_TraditionalForwardFormat
    RSYSLOG_SysklogdFileFormat
    RSYSLOG_ForwardFormat
    RSYSLOG_SyslogProtocol23Format
    RSYSLOG_DebugFormat
    RSYSLOG_WallFmt
    RSYSLOG_StdUsrMsgFmt
    RSYSLOG_StdDBFmt
    RSYSLOG_StdPgSQLFmt
    RSYSLOG_spoofadr
    RSYSLOG_StdJSONFmt
```

更多内容可以查看：

https://www.rsyslog.com/doc/v8-stable/configuration/templates.html

```
4 自定义模版
    除了使用内置模板外，我们还可以自定义模板。自定义模板可以直接写在配置文件中。
    生成的日志内容模板决定，而模板是由 rsyslog 中的相关属性组成，这些属性在生成日志内容时会被替换成具体内容。所谓属性是指 rsyslog 中的一些特殊关键字，在模板语法中，使用 %属性名% 来表示一个字段。

rsylog 常用属性
    https://www.rsyslog.com/doc/v8-stable/configuration/properties.html
    https://www.rsyslog.com/doc/v8-stable/configuration/property_replacer.html
```

Ubuntu 中日志内容

```
root@ubuntu24:~# tail /var/log/syslog
2024-10-18T14:17:05.959508+08:00 ubuntu24 systemd[1]: Starting
packagekit.service - PackageKit Daemon...
2024-10-18T14:17:05.965722+08:00 ubuntu24 PackageKit: daemon start
......

root@ubuntu24:~# tail -n 3 /var/log/auth.log
2024-10-18T14:25:16.107236+08:00 ubuntu24 sshd[1893]: pam_unix(sshd:session):
session closed for user root
2024-10-18T14:25:16.115067+08:00 ubuntu24 systemd-logind[740]: Session 3 logged
out. Waiting for processes to exit.
2024-10-18T14:25:16.121280+08:00 ubuntu24 systemd-logind[740]: Removed session
3.
......
```

Rocky | openEuler 中日志内容

```
[root@rocky9 ~]# tail /var/log/messages
Oct 18 14:43:44 rocky9 named[1177]: transfer of '0.0.10.in-addr.arpa/IN' from
10.0.0.13#53: failed to connect: connection refused
Oct 18 14:43:44 rocky9 named[1177]: transfer of '0.0.10.in-addr.arpa/IN' from
10.0.0.13#53: Transfer status: connection refused
......

[root@rocky9 ~]# tail /var/log/secure
Oct 18 09:00:19 rocky9 sshd[1081]: Server listening on :: port 22.
Oct 18 09:12:44 rocky9 sshd[1966]: Accepted password for root from 10.0.0.1 port
3385 ssh2
......
```

1.1.4 其他日志

btmp,lastlog,wtmp

```
    除了在 rsyslog 中定义的日志之外，系统中默认还有 btmp, lastlog, wtmp 三个日志文件，这三个文件都是非文本格式，无法直接打开
```

日志文件	相关命令	备注
/var/log/btmp	lastb	当前系统上，用户的失败尝试登录相关的日志 (bad logins)，二进制格式
/var/log/lastlog	lastlog	每一个用户最近一次的登录信息，二进制格式
/var/log/wtmp	last	当前系统上，用户正常登录系统的相关日志信息 (who was logged in)，二进制格式

```
在 rocky 中查看
[root@rocky9 ~]# ll /var/log/{btmp,lastlog,wtmp}
-rw-------  1 root utmp      0 May  3 14:45 /var/log/btmp
-rw-rw-r--. 1 root utmp 295212 May  3 13:59 /var/log/lastlog
-rw-rw-r--. 1 root utmp 496896 May  3 13:59 /var/log/wtmp

[root@rocky9 ~]# file /var/log/{btmp,lastlog,wtmp}
/var/log/btmp:    empty
/var/log/lastlog: data
/var/log/wtmp:    firmware 0 v0 (revision 0)   V2, 0 bytes or less, UNKNOWN2
0x6c382e78, at 0x0 0 bytes , at 0x0 0 bytes
```

```
在 ubuntu 中查看
[root@ubuntu24 ~]# ll /var/log/{btmp,lastlog,wtmp}
-rw-rw---- 1 root utmp      0 May  1 12:44 /var/log/btmp
-rw-rw-r-- 1 root utmp 292292 May  3 06:00 /var/log/lastlog
-rw-rw-r-- 1 root utmp  17664 May  3 06:00 /var/log/wtmp

[root@ubuntu24 ~]# file /var/log/{btmp,lastlog,wtmp}
/var/log/btmp:    empty
/var/log/lastlog: data
/var/log/wtmp:    data
```

命令方式查看这些特殊的日志

```
显示系统关机项和运行级别更改
last -x
last --system
```

```
dmesg 命令用来查看主机硬件相关日志
[root@rocky9 ~]# dmesg | head
[    0.000000] Disabled fast string operations
```

```
logger 命令可以手动生成相关日志
[root@rocky9 ~]# logger "this is test msg"
[root@rocky9 ~]# tail -n 1 /var/log/messages
Oct 18 15:26:25 rocky9 root[2553]: this is test msg

root@ubuntu24:~# logger "this is test msg"
root@ubuntu24:~# tail -n1 /var/log/syslog
2024-10-18T15:07:25.075254+08:00 ubuntu24 root: this is test msg
```

```
定制错误信息文件
root@ubuntu24:~# echo "error page msg" > error.txt

-t 指定日志tag, -f 从文件中读取日志内容, -p 指定优先级
root@ubuntu24:~# logger -t error -f error.txt -p error
root@ubuntu24:~# tail -n1 /var/log/syslog
2024-10-18T15:09:22.389237+08:00 ubuntu24 error: error page msg
```

1.1.5 定制日志

1.1.5.1 日志输出

基础知识

sshd服务日志

```
sshd服务日志默认是归属于 AUTH 分类，默认级别是 INFO
root@ubuntu24:~# cat /etc/ssh/sshd_config | grep LogLevel
#LogLevel INFO
```

确认rsyslog中关于auth的日志输出

```
Ubuntu系统
root@ubuntu24:~# grep '^auth' /etc/rsyslog.d/50-default.conf
auth,authpriv.*                 /var/log/auth.log

Rocky系统
[root@rocky9 ~]# grep '^auth' /etc/rsyslog.conf
authpriv.*                                              /var/log/secure
```

Ubuntu系统测试效果

```
尝试重启ssh服务
root@ubuntu24:~# systemctl restart ssh

查看日志效果
root@ubuntu24:~# tail -n 3 /var/log/auth.log
2024-10-18T15:17:01.701136+08:00 ubuntu24 CRON[4870]: pam_unix(cron:session):
session closed for user root
2024-10-18T15:19:42.086646+08:00 ubuntu24 sshd[1892]: Received signal 15;
terminating.
2024-10-18T15:19:42.147203+08:00 ubuntu24 sshd[4893]: Server listening on ::
port 22.
```

Rocky系统测试效果

```
尝试重启ssh服务
[root@rocky9 ~]# systemctl restart sshd

查看日志效果
[root@rocky9 ~]# tail -n3 /var/log/secure
Oct 18 15:39:43 rocky9 sshd[1081]: Received signal 15; terminating.
Oct 18 15:39:43 rocky9 sshd[2622]: Server listening on 0.0.0.0 port 22.
Oct 18 15:39:43 rocky9 sshd[2622]: Server listening on :: port 22.
```

1.1.5.2 定制实践

简介

```
    关于定制日志的实践，我们可以通过四个方面入手：
    1 修改服务日志的级别
    2 定制rsyslog功能日志的输出
    3 重启日志服务
    4 测试效果
```

定制sshd服务日志级别

```
1 修改 sshd 服务日志的配置项，分类改到 LOCAL6，级别不改
root@ubuntu24:~# cat /etc/ssh/sshd_config | grep -i log
# Logging
#SyslogFacility AUTH
#LogLevel INFO
SyslogFacility LOCAL6
```

```
2 新增配置文件，local6 分类的日志都写到 sshd.log 文件中
root@ubuntu24:~# cat /etc/rsyslog.d/sshd.conf
local6.*    /var/log/sshd.log
```

```
3 重启服务
root@ubuntu24:~# systemctl restart rsyslog
root@ubuntu24:~# systemctl restart ssh

查看日志文件
root@ubuntu24:~# ll /var/log/sshd.log
-rw-r----- 1 syslog adm 86 10月 18 15:26 /var/log/sshd.log

查看用户组
root@ubuntu24-13:~# grep adm /etc/group
adm:x:4:syslog,sswang
```

4 测试效果

```
操作测试
root@ubuntu24:~# ssh 127.1
root@127.0.0.1's password:
Permission denied, please try again.
root@127.0.0.1's password:

查看日志
root@ubuntu24:~# tail /var/log/sshd.log
2024-10-18T15:26:08.625210+08:00 ubuntu24 sshd[5036]: Server listening on ::
port 22.
2024-10-18T15:27:24.084924+08:00 ubuntu24 sshd[5041]: Failed password for root
from 127.0.0.1 port 36490 ssh2
2024-10-18T15:27:27.796205+08:00 ubuntu24 sshd[5041]: Accepted password for root
from 127.0.0.1 port 36490 ssh2
```

```
手工测试日志
root@ubuntu24:~# logger -p local6.info "hello sshd"

查看效果
root@ubuntu24:~# tail -1 /var/log/sshd.log
2024-10-18T15:28:16.371007+08:00 ubuntu24 root: hello sshd
```

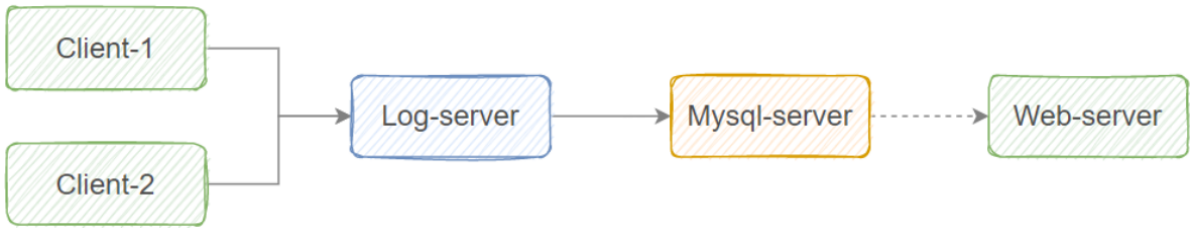
1.1.6 转发日志

1.1.6.1 角色简介

基础知识

rsyslog的角色

```
Rsyslog守护进程可以被配置成两种环境：
日志收集服务器：
    在此模式下，rsyslog进程可以从网络中收集其他主机上的日志数据，这些主机通过 TCP 或 UDP 协议会将日志配置为发送到另外的远程服务器，进行集中存储，方便统一管理。

客户端：
    在此模式下，rsyslog可以过滤和发送内部日志消息到本地文件夹（如/var/log）或一台可以路由到的远程rsyslog服务器上。
```



配置解读

```
1 客户端主机配置
    # 通过@符号，将指定的日志发送到远程主机的514端口
    *.info  @remote_addr:514

2 服务端主机配置
    # 加载udp或者tcp模块，然后开启对应的input功能
    module(load="imudp")
    input(type="imudp" port="514")
```

主机清单

主机IP	操作系统	角色
10.0.0.13	ubuntu	log-server
10.0.0.12	rocky	client-1
10.0.0.16	ubuntu	client-2

1.1.6.2 转发实践

配置 server log 主机，开启 TCP, UDP 相关功能

```
启用相关模块，去掉 UDP,TCP 模块的注释
root@ubuntu24:~# vim /etc/rsyslog.conf
...
# provides UDP syslog reception
module(load="imudp")
input(type="imudp" port="514")

# provides TCP syslog reception
module(load="imtcp")
input(type="imtcp" port="514")
```

确认接收的认证日志，转发过来的日志会被写到 /var/log/syslog 文件中

```
root@ubuntu24:~# grep auth /etc/rsyslog.d/50-default.conf
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
```

重启服务后，检查效果

```
重启服务
root@ubuntu24:~# systemctl restart rsyslog

查看端口
root@ubuntu24:~# ss -tunlp | grep 514
```

```
root@ubuntu24:~# ss -tunlp | grep 514
udp   UNCONN 0      0            0.0.0.0:514        0.0.0.0:*    users:(("rsyslogd",pid=5243,fd=5))
udp   UNCONN 0      0               [::]:514           [::]:*    users:(("rsyslogd",pid=5243,fd=6))
tcp   LISTEN 0      25           0.0.0.0:514        0.0.0.0:*    users:(("rsyslogd",pid=5243,fd=7))
tcp   LISTEN 0      25              [::]:514           [::]:*    users:(("rsyslogd",pid=5243,fd=8))
root@ubuntu24:~# netstat -tnulp | grep 514
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN      5243/rsyslogd
tcp6       0      0 :::514                  :::*                    LISTEN      5243/rsyslogd
udp        0      0 0.0.0.0:514             0.0.0.0:*                           5243/rsyslogd
udp6       0      0 :::514                  :::*                                5243/rsyslogd
root@ubuntu24:~#
```

1.1.6.3 客户端转发实践

client-1配置

配置 client-1 主机日志远程转发

```
配置远程转发，转发到 10.0.0.13 的 udp 协议514端口,默认514 可以省略
[root@rocky9 ~]# cat /etc/rsyslog.d/net.conf
*.info  @10.0.0.13:514

重启服务
[root@rocky9 ~]# systemctl restart rsyslog
```

client-1测试效果

```
测试-在client-1主机上写入日志
[root@rocky9 ~]# logger "this msg from rocky-client"

client-1 主机上也有该日志
[root@rocky9 ~]# tail -1 /var/log/messages
Oct 18 16:00:55 rocky9 root[2655]: this msg from rocky-client

在 log-server 上查看
root@ubuntu24:~# tail /var/log/syslog
2024-10-18T16:00:55+08:00 rocky9 root[2655]: this msg from rocky-client
注意：
    因为日志服务器上记录的日志非常的多，所以不一定在最后一条
```

client-2测试效果

配置 client-2 主机日志远程转发

```
配置远程转发，转发到 10.0.0.13 的 tcp 协议514端口,默认514 可以省略
root@shdns:~# cat /etc/rsyslog.d/net.conf
*.info  @@10.0.0.13:514

重启服务
root@shdns:~# systemctl restart rsyslog.service
```

client-2测试效果

```
测试
root@shdns:~# logger "this msg from ubuntu-client"

本机查看
root@shdns:~# tail -1 /var/log/syslog
2024-10-18T16:07:31.616368+08:00 shdns root: this msg from ubuntu-client

log-server 查看
root@ubuntu24:~# tail -1 /var/log/syslog
2024-10-18T16:07:31+08:00 shdns root: this msg from ubuntu-client
```

1.1.7 转储日志

1.1.7.1 逻辑解读

基础知识

操作逻辑

```
1 在将rsyslog的日志存储到MySQL之前，需要确保已经安装并配置好了MySQL服务器。
2 rsyslog客户端安装rsyslog的MySQL驱动模块
3 mysql数据库为rsyslog提供操作用户
4 rsyslog配置连接mysql数据库的配置
5 综合测试
```

主机清单

主机IP	操作系统	角色
10.0.0.16	ubuntu	mysql-server
10.0.0.13	ubuntu	log-server
10.0.0.12	rocky	client-1

1.1.7.2 mysql环境

环境部署

准备mysql服务

```
ubuntu24.04 安装mysql服务
root@ubuntu24-16:~# apt install mysql-server -y
```

允许mysql接收外部的请求

```
注释两行，不能只监听127.1
root@ubuntu24-16:~# cat /etc/mysql/mysql.conf.d/mysqld.cnf
......
#bind-address        = 127.0.0.1
#mysqlx-bind-address    = 127.0.0.1
```

重启mysql服务

```
root@ubuntu24-16:~# systemctl restart mysql.servic
```

1.1.7.3 mysql插件

在 log-server 上安装 rsyslog 的 mysql 包

```
root@ubuntu24:~# apt install rsyslog-mysql
注意：
    安装时候，提示配置mysql的连接，直接选择否
```

获取数据库相关的文件

```
查看包文件
root@ubuntu24:~# dpkg -L rsyslog-mysql
......
/usr/lib/x86_64-linux-gnu/rsyslog/ommysql.so                          #库文件
/usr/share/dbconfig-common/data/rsyslog-mysql/install/mysql           #mysql 数据表
文件
/usr/share/rsyslog-mysql/rsyslog-mysql.conf.template                  #rsyslog 配置
文件模板
......
```

查看数据库的转储模版配置

```
查看模版配置
root@ubuntu24:~# cat /usr/share/rsyslog-mysql/rsyslog-mysql.conf.template
### Configuration file for rsyslog-mysql
### Changes are preserved

module (load="ommysql")
*.* action(type="ommysql" server="_DBC_DBSERVER_" db="_DBC_DBNAME_"
uid="_DBC_DBUSER_" pwd="_DBC_DBPASS_")
```

传输数据库表文件到远程主机

```
root@ubuntu24:~# scp /usr/share/dbconfig-common/data/rsyslog-mysql/install/mysql
root@10.0.0.16:rsyslog.sql
root@10.0.0.16's password:
mysql       100% 1038   717.7KB/s   00:00
```

1.1.7.4 数据库准备

简单实践

在数据库服务器上建表

```
创建数据库
root@ubuntu24-16:~# mysql -e "create database rsyslog;"
```

```
导入SQL文件建表
root@ubuntu24-16:~# mysql -e "use rsyslog; source /root/rsyslog.sql;"
```

```
确认效果
root@ubuntu24-16:~# mysql -e "show tables from rsyslog"
+------------------------+
| Tables_in_rsyslog      |
+------------------------+
| SystemEvents           |
| SystemEventsProperties |
+------------------------+
```

在数据库服务器上授权rsysloger用户

```
创建用户并授权
mysql -e "create user 'rsysloger'@'10.0.0.%' identified by '123456';"
mysql -e "grant all on rsyslog.* to 'rsysloger'@'10.0.0.%';"
mysql -e "create user 'rsysloger'@'localhost' identified by '123456';"
mysql -e "grant all on rsyslog.* to 'rsysloger'@'localhost';"
mysql -e "flush privileges;"
```

```
查看效果
root@ubuntu24-16:~# mysql -ursysloger -p123456 -e "show tables from rsyslog"
mysql: [Warning] Using a password on the command line interface can be insecure.
+------------------------+
| Tables_in_rsyslog      |
+------------------------+
| SystemEvents           |
| SystemEventsProperties |
+------------------------+
```

1.1.7.5 rsyslog定制转储

在 log-server 服务器上配置 mysql 转发

```
准备转发规则文件
root@ubuntu24:~# cp /usr/share/rsyslog-mysql/rsyslog-mysql.conf.template
/etc/rsyslog.d/mysql.conf
```

```
修改配置
root@ubuntu24:~# cat /etc/rsyslog.d/mysql.conf
module (load="ommysql")
*.* action(type="ommysql" server="10.0.0.16" db="rsyslog" uid="rsysloger"
pwd="123456")
```

```
重启服务
root@ubuntu24:~# systemctl restart rsyslog.service
```

在 mysql 上查看是否具有相关的日志信息

```
root@ubuntu24-16:~# mysql -e "use rsyslog; select count(*) from SystemEvents;"
+----------+
| count(*) |
+----------+
|       11 |
+----------+
```

rsyslog服务器测试效果

```
测试,增加大量的日志
[root@rocky9 ~]# logger "this msg from rocky-client 1"
[root@rocky9 ~]# logger "this msg from rocky-client 2"
root@ubuntu24:~# logger "this msg from log-server"
```

```
查看数据库
root@ubuntu24-16:~# mysql -e "use rsyslog; select FromHost,Message from
SystemEvents order by id desc limit 3;"
+----------+-------------------------------+
| FromHost | Message                       |
+----------+-------------------------------+
| ubuntu24 |  this msg from log-server     |
| rocky9   |  this msg from rocky-client 2 |
| rocky9   |  this msg from rocky-client 1 |
+----------+-------------------------------+
```

1.2 日志工具

1.2.1 journalctl

1.2.1.1 工具简介

基础知识

简介

```
    在 systemd 为 1号进程的系统版本中，systemd 提供了一个集中的方式来处理所有来自进程，应用程序等的操作系统日志，所有这些日志事件都由 systemd 的 journald 守护进程来处理。journald 守护进程收集所有来自 Linux 操作系统的各种日志，并将其作为二进制数据存储在文件中。
```

```
    以二进制数据集中记录事件、系统问题的好处有很多。例如，由于系统日志是以二进制而不是文本形式存储的，你可以 以文本、JSON 对象等多种方式进行转译，以满足各种需求。另外，由于日志是按顺序存储的，通过对日志的日期/时间操作，超级容易追踪到单个事件。
```

```
也就是说，默认情况下，系统的日志一般会存两份：
    一份是文本的，由rsyslog去存
    一份是二进制的，由journald去存，由journctl去管理
```

文件现状

```
查看文件大小
root@ubuntu24:~# du -sh /var/log/journal/
198M    /var/log/journal/

查看文件权限
root@ubuntu24:~# ll /var/log/journal/a839fdf2a0f54609a3f4a0c9283f3375/
drwxr-sr-x+ 2 root systemd-journal 4096 11月 26 21:50 ./
drwxr-sr-x+ 3 root systemd-journal 4096 11月 26 21:50 ../
结果显示：
    这里用到了SGID的权限，也就是说，该目录下的文件，无论是谁创建的都是一个用户组下的权限
    + 表明这里还用到了ACL的权限

查看扩展的特殊权限
root@ubuntu24-13:~# getfacl /var/log/journal/a839fdf2a0f54609a3f4a0c9283f3375/
getfacl: 从绝对路径名尾部去除" / "字符。
# file: var/log/journal/a839fdf2a0f54609a3f4a0c9283f3375/
# owner: root
# group: systemd-journal
# flags: -s-
```

```
user::rwx
group::r-x
group:adm:r-x
mask::r-x
other::r-x
default:user::rwx
default:group::r-x
default:group:adm:r-x
default:mask::r-x
default:other::r-x
```

查看文件内容类型

```
root@ubuntu24:~# file
/var/log/journal/a839fdf2a0f54609a3f4a0c9283f3375/system@000624538fad8ad4-
52eff14d2a09fdc5.journal~
/var/log/journal/a839fdf2a0f54609a3f4a0c9283f3375/system@000624538fad8ad4-
52eff14d2a09fdc5.journal~: Journal file, Sat Oct 12 06:45:32 2024, online, keyed
hash siphash24, compressed zstd, compact, header size 0x110, entries 0xa31
```

日志数据存储目录，非文本文件，所以"无法"用文本工具打开

```
root@ubuntu24:~# tree /var/log/journal/
```

```
root@ubuntu24:~# tree /var/log/journal/
/var/log/journal/
└── a839fdf2a0f54609a3f4a0c9283f3375
    ├── system@000624538fad8ad4-52eff14d2a09fdc5.journal~
    ├── system@000624656aab2a27-8b3acc1238321f4d.journal~
    ├── system@00062466930b37ca-5e75606bb9daa04b.journal~
    ├── system@00062479827c464c-157012db01e27c53.journal~
    ├── system@00062485f9f381e2-869aeb85fb3e8c39.journal~
    ├── system@0006248d81f1664e-2a62e9b20becbb52.journal~
    ├── system@00062497477901c8e-47bc5d423c98be7f.journal~
    ├── system@00062498dade81ee-04838a504a0f5b7f.journal~
    ├── system@000624a1a9b7aab1-d2c41eab6e4ccad4.journal~
    ├── system@000624b5d2f80d53-91b5761ba194e2c3.journal~
    ├── system@000624b62afda865-3d69b115d97b8a49.journal~
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000cc43-000624a1a9a780ce.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000da36-0007439ff9e4ac6e.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dc7a-000624a6bc2c93ef.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dc81-000743a074d5e7c5.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dccf-000624a6c09dc97e.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dcd1-000743a0755a0ee2.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dcfb-000624a6c121b48d.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dd42-00061cc105183114.journal
    ├── system@1934e5d60ba94def80c038158d09af88-000000000000dd46-000614e55d2b98a3.journal
    └── system@2f7ccd0769dd40b589be6b21eecfba3f-000000000000eb5c-000624b62aed18ed.journal
```

Rocky系统

在CentOS 7及其后续版本中，systemd-journald是一个改进型的日志管理服务，它会收集来自内核、系统早期启动阶段的日志、系统守护进程在启动和运行中的标准输出和错误信息，以及syslog的日志。默认情况下，systemd-journald并不会持久化保存日志到/var/log/journal/目录（除非进行了特定配置），而是保存在/run/log/下的某个目录中，且日志文件是压缩和格式化的二进制数据。

如果希望将systemd-journald的日志持久化保存到/var/log/journal/目录，可以通过修改/etc/systemd/journald.conf配置文件中的Storage=选项来实现。例如，将Storage=auto或transient更改为Storage=persistent，并创建/var/log/journal/目录后，systemd-journald就会在下次启动时将日志写入该目录。

journalctl工具

journalctl是Linux系统中一个用于查看和管理日志信息的命令行工具，它是systemd日志记录系统（systemd-journald服务）的一部分。

基本功能

查看日志：
    journalctl允许用户查看整个系统的日志，或者特定服务的日志。日志信息默认存储在内存中，并且可以选择性地保存到磁盘。

过滤日志：
    journalctl提供了丰富的过滤选项，用户可以根据时间、服务单元、优先级等多种条件来过滤日志，从而快速定位到所需的日志信息。

实时监控：
    类似于tail -f命令，journalctl支持实时查看日志更新，即可以在命令执行过程中动态更新日志。

1.2.1.2 命令解读

命令解读

```
命令格式：
    journalctl [OPTIONS...] [MATCHES...]
```

```
常用选项
    -u|--unit=UNIT                  #根据单元查看日志
    -o|--output=STRING              #指定输出格式，默认 short
    -b|--boot[=ID]                  #查看详细启动日志
    --list-boots                    #列出系统启动日志
    -n 2                            #限制日志条目数量
    -x                              #显示详细的日志信息，包括与日志条目相关的所有上下文信息
    -e                              #显示的时候，直接跳转到内容的末尾
```

```
一般选项
    --system                        #显示系统服务和内核相关日志
    --user                          #显示来自当前用户的服务日志
    -M|--machine=CONTAINER          #显示本地容器中的日志
    -S|--since=DATE                 #显示从指定时间之后的日志，YYYY-MM-DD HH:MM:SS|yesterday|today|tomorrow|now
    ...
    --header                        #显示日志元数据
    --list-catalog                  #简略显示日志分类信息
    --dump-catalog                  #详细显示日志分类信息
    --update-catalog                #更新日志分类索引二进制文件
```

输出格式说明

```
    short                       #默认值，每行一条日志
    short-precise               #时间精确到微秒显示
    short-iso                   #以ISO 8601格式显示时间
    short-iso-precise           #将时间戳字段的零值从内核启动时开始计算
    short-full                  #与 short 内容相同，但时间显示更详细
    short-monotonic             #将时间戳字段的零值从内核启动时开始计算
    short-unix                  #将时间戳字段显示为从"UNIX时间原点"(1970-1-1 00:00:00UTC)以来的秒数。 精确到微秒级别
    verbose                     #以结构化的格式显示每条日志的所有字段
    export                      #将日志序列化为二进制字节流(大部分依然是文本) 以适用于备份与网络传输
    json                        #将日志项按照JSON数据结构格式化， 每条日志一行
```

```
    json-pretty                 #将日志项按照JSON数据结构格式化， 但是每个字段一行， 以便于人类阅读
    json-sse                    #将日志项按照JSON数据结构格式化，每条日志一行，但是用大括号包围
    json-seq                    #将日志项按照JSON数据结构格式化，每条日志前面加上ASCII记录分隔符，后面加上ASCII换行符
    cat                         #仅显示日志的实际内容， 而不显示与此日志相关的任何元数据(包括时间戳)
    with-unit                   #类似于 short-full,前缀以unit 名来显示，而不是syslog标识符
```

常用组合

```
journalctl -xeu myservice.service
```

简单实践

范例：查看所有日志

```
默认显示所有日志
root@ubuntu24:~# journalctl
10月 12 10:47:01 sswang kernel: last_pfn = 0x140000 max_arch_pfn = 0x400000000
......

root@ubuntu24:~# ll /var/log/syslog*
-rw-r----- 1 syslog adm 1450696 Jun 30 11:50 /var/log/syslog
-rw-r----- 1 syslog adm 1178030 Jun 26 13:56 /var/log/syslog.1

内容和 syslog 中一样
root@ubuntu24:~# grep '0x400000000' /var/log/syslog.1
2024-10-18T09:24:47.716260+08:00 ubuntu24 kernel: last_pfn = 0x140000
max_arch_pfn = 0x400000000

但删除掉 syslog ，journalctl 还是能查看
root@ubuntu24:~# mv /var/log/sys* /tmp/
root@ubuntu24:~# journalctl
10月 12 10:47:01 sswang kernel: last_pfn = 0x140000 max_arch_pfn = 0x400000000
......
```

1.2.1.3 信息过滤

常用参数实践

```
根据服务或程序查看
root@ubuntu24:~# journalctl -u nginx.service
root@ubuntu24:~# journalctl -u ssh.service
root@ubuntu24:~# journalctl /usr/sbin/sshd
root@ubuntu24:~# journalctl /usr/bin/bash
root@ubuntu24:~# journalctl -b -u nginx.service -o json-pretty
```

查看特定范围日志

```
日志跟随，显示最新日志
root@ubuntu24:~# journalctl -f

倒序显示
root@ubuntu24:~# journalctl -r
```

```
查看 2023-10-18 12:0:0 到 2024-10-18 15:30:0 之间产生的日志
root@ubuntu24:~# journalctl -S "2023-10-18 12:0:0" -U "2024-10-18 15:30:0"

查看昨天到当前的日志
root@ubuntu24:~# journalctl -S yesterday

查看前100小时到1小时之前的日志
root@ubuntu24:~# journalctl --since "100 hour ago" --until "1 hour ago"
```

查看特定信息

```
查看内核日志
root@ubuntu24:~# journalctl -k

查询日志占用了多少磁盘空间
root@ubuntu24:~# journalctl --disk-usage
Archived and active journals take up 80.0M in the file system.
```

```
查看启动信息
root@ubuntu24:~#
-7 d1517808b2da49d5a6a42b53e625c0f5 Thu 2023-05-04 17:02:23 CST-Thu 2023-05-04
21:03:47 CST
...
属性条目解读：
    第一列是每次启动序号                    第二列是启动ID，32位
    第三列是启动后的系统运行时长             最后一条是最新的数据
    -7 是代表是倒数第七次

查看本次启动日志
root@ubuntu24:~# journalctl -b 0

查看上次启动日志
root@ubuntu24:~# journalctl -b -1
root@ubuntu24:~# journalctl -b d1517808b2da49d5a6a42b53e625c0f5
```

```
root@ubuntu24:~# journalctl --list-boots
IDX BOOT ID                          FIRST ENTRY                 LAST ENTRY
-12 ac639c4dcaf348b4b93bfdf8ce4dcf6d Sat 2024-10-12 10:47:01 CST Sat 2024-10-12 14:44:22 CST
-11 8f2b98abd24f45d5b95cc186b29d993c Sat 2024-10-12 14:44:36 CST Sat 2024-10-12 23:17:01 CST
-10 bf11eeda887e4572a19a22773a1aa9db Sun 2024-10-13 11:46:10 CST Mon 2024-10-14 00:00:57 CST
 -9 9586143f25c44526b74378ac1190f7be Mon 2024-10-14 09:04:19 CST Mon 2024-10-14 10:20:31 CST
 -8 1ca13e254f2e47c8bc516e2822ea1969 Mon 2024-10-14 10:27:11 CST Mon 2024-10-14 18:20:14 CST
 -7 daba5db528344c3383309f5c7dc32f57 Tue 2024-10-15 09:02:38 CST Tue 2024-10-15 23:53:02 CST
 -6 226b4d576d2e4879a208feb51e43c30e Tue 2024-10-15 23:55:02 CST Wed 2024-10-16 00:10:48 CST
 -5 6df4b1fdb2cf41ff9f275c2da8df0168 Wed 2024-10-16 08:54:08 CST Wed 2024-10-16 15:30:59 CST
 -4 25962c2b3ecc4b0f8189a677e3216aa5 Wed 2024-10-16 17:12:18 CST Wed 2024-10-16 18:45:38 CST
 -3 7a7a6a154cc542fea076f41dffd98ba4 Wed 2024-10-16 22:26:24 CST Thu 2024-10-17 00:07:01 CST
 -2 3820fdc49d5847b196e0be320edcf173 Thu 2024-10-17 08:56:54 CST Thu 2024-10-17 15:54:55 CST
 -1 513b93a7b6de428fb369e1d4a7790dbf Fri 2024-10-18 09:24:43 CST Fri 2024-10-18 17:15:01 CST
  0 c4f7a5e1f9ad47aab13f7aaea19942cc Fri 2024-10-18 09:00:06 CST Wed 2034-10-18 09:23:37 CST
```

1.2.1.4 其他属性实践

用户数据参数实践

```
根据PID查看
root@ubuntu24:~# journalctl _PID=1

根据UID查看
root@ubuntu24:~# journalctl _UID=0
```

日志级别

```
根据等级查看
root@ubuntu24:~# journalctl -p warning
root@ubuntu24:~# journalctl -p 4
注意：
    -p 是 --priority 的缩写，后面跟的数字表示日志的优先级或级别。

根据分类查看
root@ubuntu24:~# journalctl --facility=auth
```

日志显示格式和条目限制

```
指定显示格式， -n 1 这个选项用于限制显示的日志条目数量。
root@ubuntu24:~# journalctl -o json-pretty -n 1
root@ubuntu24:~# journalctl -o cat -n 1

显示特定字段
root@ubuntu24:~# journalctl -o json-pretty -n 1 --output-fields=_UID
```

文件属性信息查看实践

```
查看日志文件元数据
root@ubuntu24:~# journalctl --header
......
注意：
    该命令默认会查看所有的文件元数据
```

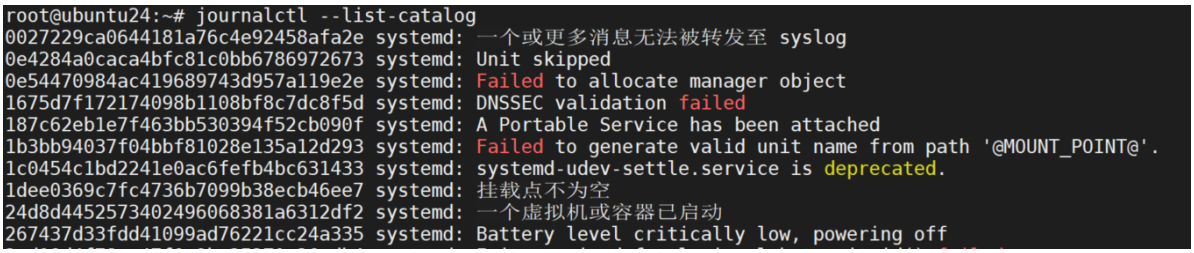
```
root@ubuntu24:~# journalctl --header
File path: /var/log/journal/a839fdf2a0f54609a3f4a0c9283f3375/system@000624a1a9b7aab1-d2c41eab6e4ccad4.journal~
File ID: 5f3101d7ccc542dcac1bce4575c4b15e
Machine ID: a839fdf2a0f54609a3f4a0c9283f3375
Boot ID: 7a7a6a154cc542fea076f41dffd98ba4
Sequential number ID: f38fac219d9f4dfea12ac21e375235c1
State: ONLINE
Compatible flags: TAIL_ENTRY_BOOT_ID
Incompatible flags: COMPRESSED-ZSTD KEYED-HASH COMPACT
Header size: 272
Arena size: 8388336
Data hash table size: 233016
Field hash table size: 333
Rotate suggested: no
Head sequential number: 51858 (ca92)
Tail sequential number: 52290 (cc42)
Head realtime timestamp: Thu 2024-10-17 00:04:06 CST (6249a38373907)
Tail realtime timestamp: Thu 2024-10-17 00:07:01 CST (6249a42a26705)
Tail monotonic timestamp: 1h 40min 48.091s (1687e8eb5)
Objects: 895
Entry objects: 113
```

查看日志分类

第一列为分类ID
```
root@ubuntu24:~# journalctl --list-catalog
0027229ca0644181a76c4e92458afa2e systemd: 一个或更多消息无法被转发至 syslog
...
```

显示指定分类信息
```
root@ubuntu24:~# journalctl --list-catalog 0027229ca0644181a76c4e92458afa2e
0027229ca0644181a76c4e92458afa2e systemd: 一个或更多消息无法被转发至 syslog
```

详细显示分类信息
```
root@ubuntu24:~# journalctl --dump-catalog 0027229ca0644181a76c4e92458afa2e
-- 0027229ca0644181a76c4e92458afa2e
Subject: 一个或更多消息无法被转发至 syslog
Defined-By: systemd
Support: http://www.ubuntu.com/support

有一条或更多的消息无法被转发至与 journald 同时运行的 syslog 服务。
这通常意味着 syslog 实现无法跟上队列中消息进入的速度。
```



1.2.2 Logrotate

1.2.2.1 工具简介

基础知识

简介

> 在 Linux 系统中，能够帮助使用者定位问题的有效手段之一就是查日志。
>
> 如果一个服务或一个程序的日志，一直只写一个文件，则会导致该日志文件越来越大，无论是查看还是搜索内容，备份等，都会特别不方便，而且如果服务器数量较多，日志文件大小增长较快，也会很容易触发告警。
>
> 为了解决这种情况，我们可以使用日志转储服务，对服务日志进行分割，按照一定的规则将日志保存在不同的文件中，这样更便于管理和归档。

当前服务器上的日志转储

```
root@ubuntu24:~# ll /var/log/dmesg*
-rw-r----- 1 root adm 132571 Jul  4 08:49 /var/log/dmesg
-rw-r----- 1 root adm 132565 Jul  3 20:12 /var/log/dmesg.0
-rw-r----- 1 root adm  26381 Jun 30 08:18 /var/log/dmesg.1.gz
-rw-r----- 1 root adm  26443 Jun 28 22:01 /var/log/dmesg.2.gz

root@ubuntu24:~# ll /var/log/alternatives.log*
-rw-r--r-- 1 root root  416 Jul  4 08:49 /var/log/alternatives.log
-rw-r--r-- 1 root root 9415 Jun 30 08:31 /var/log/alternatives.log.1
-rw-r--r-- 1 root root 3160 May  9 11:58 /var/log/alternatives.log.2.gz
```

工具简介

logrotate简介

```
    logrotate 程序是一个日志文件管理工具。用来把旧的日志文件删除，并创建新的日志文件，称为日志转储或滚动。可以根据日志文件的大小，也可以根据其天数来转储，这个过程一般通过 cron 程序来执行。
```

各Linux 发行版会默认安装 logrotate 包

```
华为欧拉系统：
[root@openEuler ~]# rpm -q logrotate
logrotate-3.21.0-1.oe2403.x86_64

Rocky9系统
[root@rocky9 ~]# rpm -q logrotate
logrotate-3.18.0-8.el9.x86_64

ubuntu系统
root@ubuntu24:~# dpkg -l logrotate
```

```
root@ubuntu24:~# dpkg -l logrotate
期望状态=未知(u)/安装(i)/删除(r)/清除(p)/保持(h)
| 状态=未安装(n)/已安装(i)/仅存配置(c)/仅解压缩(U)/配置失败(F)/不完全安装(H)/触发器等待(W)/触发器未决(T)
|/ 错误?=(无)/须重装(R) (状态，错误：大写=故障)
||/ 名称           版本            体系结构     描述
+++-==============-===============-============-=================================
ii  logrotate      3.21.0-2build1 amd64        Log rotation utility
root@ubuntu24:~#
```

相关文件

```
root@ubuntu24:~# dpkg -S logrotate
/etc/cron.daily/logrotate           #定时任务脚本，放在 cron.daily 目录中，默认系统会每天执行一次
/etc/logrotate.conf                 #主配置文件，定义日志转储策略
/etc/logrotate.d/                   #配置文件目录，定义日志转储策略
/usr/sbin/logrotate                 #主程序
/var/lib/logrotate/status           #logrotate服务的日志文件
```

```
系统计划任务每天执行一次脚本文件，即调用 logrotate 程序再配合定义好的转储规则对日志文件进行转储。
root@ubuntu24:~# cat /etc/cron.daily/logrotate
#!/bin/sh

# skip in favour of systemd timer
# 如果存在该目录，这意味着系统正在使用 systemd 进行服务管理。
# 由于 systemd 可以设置定时器来运行 logrotate，因此如果检测到 systemd，脚本将不会继续执行
```

```
# 而是直接退出（返回状态码 0，表示成功退出但不执行任何操作）。
if [ -d /run/systemd/system ]; then
    exit 0
fi

# this cronjob persists removals (but not purges)
# 如果 logrotate 不可执行，脚本同样会退出（返回状态码 0）。
# 这个检查确保了只有在 logrotate 可用时，脚本才会尝试运行它。
if [ ! -x /usr/sbin/logrotate ]; then
    exit 0
fi

# 基于配置进行日志切割动作
/usr/sbin/logrotate /etc/logrotate.conf
EXITVALUE=$?
if [ $EXITVALUE != 0 ]; then
    /usr/bin/logger -t logrotate "ALERT exited abnormally with [$EXITVALUE]"
fi
exit $EXITVALUE
```

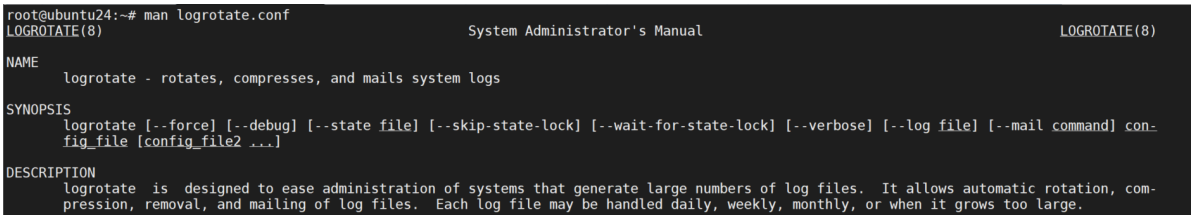
1.2.2.2 配置解读

常见的配置文件

在配置文件中定义转储规则，配置文件中的主要配置项

查看帮助

```
root@ubuntu24:~# man logrotate.conf
```



```
root@ubuntu24:~# man logrotate.conf
LOGROTATE(8)                                System Administrator's Manual                                LOGROTATE(8)

NAME
       logrotate - rotates, compresses, and mails system logs

SYNOPSIS
       logrotate [--force] [--debug] [--state file] [--skip-state-lock] [--wait-for-state-lock] [--verbose] [--log file] [--mail command] con-
       fig_file [config_file2 ...]

DESCRIPTION
       logrotate  is  designed to ease administration of systems that generate large numbers of log files.  It allows automatic rotation, com-
       pression, removal, and mailing of log files.  Each log file may be handled daily, weekly, monthly, or when it grows too large.
```

主配置文件

```
root@ubuntu24:~# cat /etc/logrotate.conf
weekly                                                  #默认每周一次转储
su root adm                                             #默认使用 adm 组
rotate 4                                                #默认保留最近4周的文件(4个文
件)
create                                                  #转储完成后生成新的空文件
#dateext                                                #默认不使用日期后缀
#compress                                               #默认不启用文件压缩
include /etc/logrotate.d                                #包含的子目录
```

每个服务单独的配置文件，如果在单独配置文件中没有定义的配置项，则使用主配置文件中的配置项或默认配置

查看子配置文件

```
root@ubuntu24:~# ls /etc/logrotate.d/
alternatives  apport  apt  bootlog  btmp  dbconfig-common  dpkg  nginx  php8.1-
fpm  rsyslog  ubuntu-advantage-tools  ufw  unattended-upgrades  wtmp
```

查看日志切割规则

```
root@ubuntu24:~# cat /etc/logrotate.d/rsyslog
/var/log/syslog
...
/var/log/messages
{                       #上述所有日志文件都适用于此转储规则
    rotate 4            #保留最近4个文件，加上当前使用的，一个5个
    weekly              #每周转储
    missingok           #如果要转储的日志文件不存在，不提示错误，继续下一个
    notifempty          #如果是空文件，不转储
    compress            #启用gzip压缩转储后的日志文件
    delaycompress       #和 compress 一起使用时，转储的日志文件到下一次转储时才压缩
    sharedscripts       #运行脚本，分别是转储前和转储后脚本
    postrotate          #转储后脚本
        /usr/lib/rsyslog/rsyslog-rotate
    endscript
}
```

logrotate规则格式解读

```
/path/to/logfile {      # /path/to/logfile表示待管理的日志文件路径，必须指定其绝对路径；
    option1 value1
    option2 value2
    ...
}
解析：
    option1、option2等表示logrotate的选项
    value1、value2等表示选项的值。
```

配置项解读

常用配置项

```
rotate count                    #指定日志文件删除之前转储的次数，0 指没有备份，5 指保留5
个备份
create mode owner group         #转储文件，使用指定的权限，所有者，所属组创建新的日志文件
compress                        #通过gzip压缩转储以后的日志
delaycompress                   #和 compress 一起使用时，转储的日志文件到下一次转储时才
压缩
prerotate/endscript             #在转储之前需要执行的命令
postrotate/endscript            #在转储以后需要执行的命令，关键字必须单独成行，后面是日志
文件路径
                                #可以出现多个脚本
daily                           #指定转储周期为每天
weekly                          #指定转储周期为每周
monthly                         #指定转储周期为每月
missingok                       #如果日志不存在，不提示错误，继续处理下一个
size size                       #当日志文件到达指定的大小时才转储，默认单位是bytes，也可
是KB或MB
```

一般配置项

```
nocompress                      #不压缩
copytruncate                    #用于还在打开中的日志文件，把当前日志备份并截断
nocopytruncate                  #用于还在打开中的日志文件，把当前日志备份并不截断
nocreate                        #不建立新的日志文件
su user group                   #指定转储的用户和组，
```

```
                                #如果日志文件的父目录属组或other 具有写权限，则要指定此处
nodelaycompress                 #覆盖 delaycompress 选项，转储同时压缩
errors address                  #转储时的错误信息发送到指定的Email 地址
ifempty                         #即使是空文件也转储，此为默认选项
notifempty                      #如果是空文件的话，不转储
mail address                    #把转储的日志文件发送到指定的E-mail 地址，依赖邮件服务
nomail                          #转储时不发送日志文件
olddir directory                #转储后的日志文件放入指定目录，必须和当前日志文件在同一个
文件系统
noolddir                        #转储后的日志文件和当前日志文件放在同一个目录下
tabooext [+] list               #让logrotate不转储指定扩展名的文件，
                                #缺省的扩展名是：.rpm-orig,.rpmsave, v, 和 ~
sharedscripts                   #对每个转储日志运行prerotate和postrotate脚
nosharedscripts                 #针对每一个转储的日志文件，
                                #都执行一次prerotate 和 postrotate脚本，此为默认值
nomissingok                     #如果日志不存在，提示错误，此为默认值
```

compress + delaycompress 指的是 压缩，但是最近一次不压缩

```
转储前
root@ubuntu24-13:~# ls /var/log/syslog*
/var/log/syslog
/var/log/syslog.1
/var/log/syslog.2.gz
```

```
转储后
                    -------------  syslog
/var/log/syslog     -------------  syslog.1
/var/log/syslog.1   -------------  syslog.2.gz
/var/log/syslog.2.gz -------------  syslog.3.gz

delaycompress 的作用是第一次转储不压缩，第二次转储进行压缩
rotate 4 的作用是报存4个，超过4个就删除，也就是说，一旦出现syslog.4.gz 就删除
```

1.2.2.3 规则实践

命令解读

logrotate 命令

```
命令格式
    logrotate [OPTION...] <configfile>

常用选项
    # 没有什么常用选项，直接使用默认选项

一般选项
-?|--help                       #显示帮助信息
-d|--debug                      #不执行任何操作，仅显示错误信息，类似于测试
-f|--force                      #强制执行
-m|--mail=command               #指定执行邮件发送的命令，默认 /usr/bin/mail
-s|--state=statefile            #指定服务日志文件，默认 /var/lib/logrotate/status
-v|--verbose                    #显示详细信息
-l|--log=logfile                #指定详细信息日志，加上此选项，会将详细信息写到指定文件
```

```
--version                       #显示版本信息
--skip-state-lock               #不给 statefile 文件加锁
```

简单实践

准备工作

```
创建测试日志文件
root@ubuntu24:~# dd if=/dev/zero of=/var/log/test1.log bs=2M count=1
root@ubuntu24:~# dd if=/dev/zero of=/var/log/test2.log bs=2M count=1

查看文件效果
root@ubuntu24:~# ll -h /var/log/test*
-rw-r--r-- 1 root root 2.0M 10月 18 18:10 /var/log/test1.log
-rw-r--r-- 1 root root 2.0M 10月 18 18:10 /var/log/test2.log
```

自定义规则

```
定义转储规则
root@ubuntu24:~# cat /etc/logrotate.d/test{1,2}
/var/log/test1.log {
  daily
  rotate 5
  su root root
  compress
  delaycompress
  missingok
  size 1M
  notifempty
  create 0640 syslog adm
  postrotate
     echo `date +%F_%T` >> /tmp/test1.log
  endscript
}

/var/log/test2.log {
  daily
  rotate 5
  su root root
  dateext
  compress
  delaycompress
  missingok
  size 1M
  notifempty
  create 644 root root
  postrotate
    echo `date +%F_%T` >> /tmp/test2.log
  endscript
}
注意：
    test2的后缀使用日期格式
```

执行转储测试

```
方法1： 手动执行转储
root@ubuntu24:~# logrotate /etc/logrotate.d/test1

查看日志，生成新的空文件，权限，属主属组都符合预设
root@ubuntu24:~# ls -lh /var/log/test1*
-rw-r----- 1 syslog adm     0 10月 18 18:11 /var/log/test1.log
-rw-r--r-- 1 root   root 2.0M 10月 18 18:10 /var/log/test1.log.1
```

```
先保证日志达到转储条件
root@ubuntu24:~# dd if=/dev/zero of=/var/log/test1.log bs=3M count=1

再次转储
root@ubuntu24:~# logrotate /etc/logrotate.d/test1

查看效果
root@ubuntu24:~# ls -lh /var/log/test1*
-rw-r----- 1 syslog adm     0 10月 18 18:12 /var/log/test1.log
-rw-r----- 1 syslog adm  3.0M 10月 18 18:12 /var/log/test1.log.1     #最新的转储
-rw-r--r-- 1 root   root 2.1K 10月 18 18:10 /var/log/test1.log.2.gz  #前一个被转储
的日志被压缩

查看日志文件，转储成功后命令被执行
root@ubuntu24:~# cat /tmp/test1.log
2024-10-18_18:11:43
2024-10-18_18:12:19
```

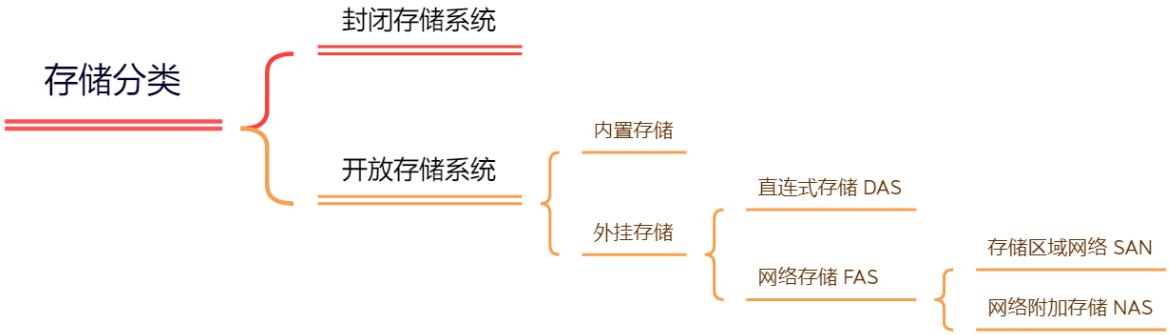
```
方法2：调用主配置文件进行转储，主配置文件中包含了 test1 test2 配置，所以都会被转储
root@ubuntu24:~# logrotate /etc/logrotate.conf

test1 没有达到转储条件，所以不发生改变
root@ubuntu24:~# ls -lh /var/log/test1*
-rw-r----- 1 syslog adm     0 10月 18 18:12 /var/log/test1.log
-rw-r----- 1 syslog adm  3.0M 10月 18 18:12 /var/log/test1.log.1
-rw-r--r-- 1 root   root 2.1K 10月 18 18:10 /var/log/test1.log.2.gz

test2 达到条件，且有日期后缀
root@ubuntu24:~# ls -lh /var/log/test2*
-rw-r--r-- 1 root root    0 10月 18 18:17 /var/log/test2.log
-rw-r--r-- 1 root root 2.0M 10月 18 18:10 /var/log/test2.log-20241018
```

1.3 共享存储

1.3.1 存储类型

1.3.1.1 常见存储

存储分类



开放存储系统

开放存储系统采用高速网络，把多套PC服务器连接起来，构建一个开放式系统，这些PC服务器称作存储节点。它提供了可扩展的接口，方便安装软件以增强系统的功能。

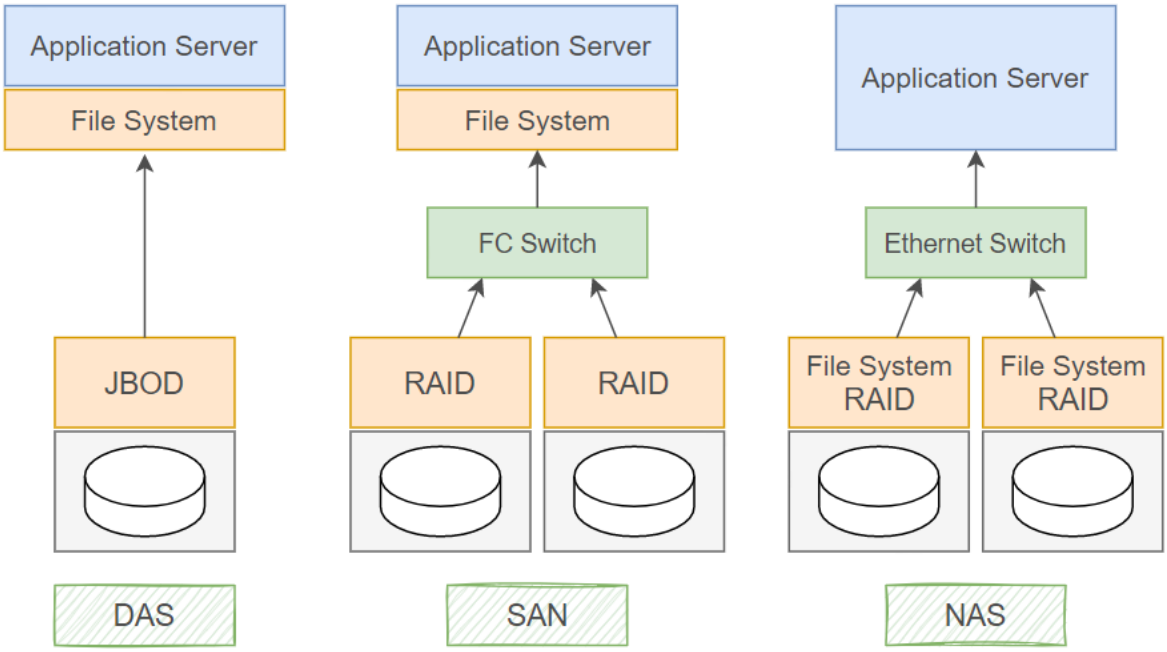
开放存储系统适用于需要大规模存储、高可扩展性和灵活性的场景，如云计算、大数据处理等领域。

封闭存储系统

封闭存储系统主要指大型机或专用存储设备的存储架构，这些系统在设计时已经明确其用途和使用功能情景，不提供扩展接口来扩展软件功能。

封闭存储系统适用于对性能和稳定性要求极高、且不需要频繁扩展存储容量的场景，如金融、电信等行业的关键业务系统。

存储类型



直连式存储：Direct-Attached Storage（DAS）

可以理解为本地文件系统。这种设备直接连接到计算机主板总线上，通常是SCSI 或 FC 连接，计算机将其识别为一个块设备，例如常见的硬盘，U盘等，也可以连到一个磁盘阵列柜，里面有很多块磁盘。 DAS 只能连接一台服务器，其它服务器无法共享该存储。

存储区域网络：Storage Area Network（SAN）

存储区域网络，这个是通过光纤通道或以太网交换机连接存储阵列和服务器主机，最后成为一个专用的存储网络。SAN经过十多年历史的发展，已经相当成熟，成为业界的事实标准（但各个厂商的光纤交换技术不完全相同，其服务器和SAN存储有兼容性的要求）。

SAN提供了一种与现有LAN连接的简易方法，并且通过同一物理通道支持广泛使用的SCSI和IP协议。SAN不受现今主流的、基于SCSI存储结构的布局限制。特别重要的是，随着存储容量的爆炸性增长，SAN允许企业独立地增加它们的存储容量。SAN的结构允许任何服务器连接到任何存储阵列，这样不管数据置放在那里，服务器都可直接存取所需的数据。因为采用了光纤接口，SAN还具有更高的带宽。

如今的SAN解决方案通常会采取以下两种形式：光纤信道以及iSCSI或者基于IP的SAN，也就是FC SAN和IP SAN。光纤信道是SAN解决方案中大家最熟悉的类型，但是，最近一段时间以来，基于iSCSI（Internet SCSI）或FCIP（Fibre Channel over IP）的SAN解决方案开始大量出现在市场上，与光纤通道技术相比较而言，这种技术具有良好的性能，存储设备可以被分配IP地址，并通过IP网络进行数据传输，而且价格低廉。一般通过在主机上安装HBA(Host Bus Adapter 主机总线适配器)卡，再通过光纤连接到光纤交换机,再连接至SAN存储上。这使得存储设备能够跨越不同的物理位置，通过网络进行远程访问和管理。

在存储区域网络（SAN）中，连接的设备是否拥有自己的IP地址取决于具体的网络架构和技术实现。传统的SAN架构，如基于光纤通道的SAN，通常不直接为存储设备分配IP地址。在这种架构中，存储设备和服务器之间的通信是通过专用的存储协议（如光纤通道协议）来实现的，这些协议在物理层和数据链路层上工作，并不涉及IP层。因此，传统的SAN设备通常没有自己的IP地址。

网络附加存储：Network-Attached Storage (NAS)

NAS存储也通常被称为附加存储，顾名思义，就是存储设备通过标准的网络拓扑结构(例如以太网)添加到一群计算机上。与DAS以及SAN不同，NAS是文件级的存储方法。采用NAS较多的功能是用来进行文件共享。而且随着云计算的发展，一些NAS厂商也推出了云存储功能，大大方便了企业和个人用户的使用。NAS产品是真正即插即用的产品。

NAS设备一般支持多计算机平台，用户通过网络支持协议可进入相同的文档，因而NAS设备无需改造即可用于混合Unix/Windows NT局域网内，同时NAS的应用非常灵活。

三种存储类型比较

类型	传输类型	数据类型	典型应用	优点	缺点
DAS	SCSI、FC	数据块	任何	磁盘与服务器分离，便于管理	连接距离短，数据分散，共享困难，存储利用率不高，扩展性有限
NAS	IP	文件	文件服务器	不占用服务器资源，即插即用	不适合存储量大的块级应用数据，备份及恢复占用网络带宽
SAN	IP、FC、SAS	数据块	数据库服务器	高扩展性|可用性，易管理	相比NAS成本较高，安装和升级比NAS复杂

应用场景
DAS：适用于那些数据量不大，对磁盘访问速度要求较高的中小企业
NAS：多适用于文件服务器，用来存储非结构化数据，虽然受限于以太网的速度，但是部署灵活，成本低
SAN：适用于大型应用或数据库系统，缺点是成本高、较为复杂

1.3.1.2 存储方式

块存储

> 块存储（Block Storage Service）设备不能被操作系统直接访问，需要创建分区或逻辑卷，再创建文件系统（EXT3，EXT4，NTFS 等），然后再经过挂载，才能使用。对于主机而言，其并不能识别出存储设备是真正的物理设备还是二次划分的逻辑设备（RAID等），块存储不仅仅是直接使用物理设备，间接使用物理设备的也叫块设备。比如虚拟机上创建的虚拟磁盘。
>
> 块存储大多数时候是本地存储，读写速度很快，但不利于扩展，数据不能被共享。

文件存储

> 文件存储（File Storage Service）可以分为本地文件存储和网络文件存储。文件存储最明显的特征是支持POSIX的文件访问接口，例如 open、read、write、seek、close 等（这几个是常用的文件操作API接口 ），与块存储不同，主机不需要再对文件存储进行格式化和创建文件系统，文件管理功能已由文件存储自行管理。
>
> 文件存储的特点是便于扩展和共享，但读写速度较慢。

对象存储

> 对象存储（Object Storage Service）也叫基于对象的存储，是一种解决和处理离散单元的方法，可提供基于分布式系统之上的对象形式的数据存储服务。对象存储和我们经常接触到的块和文件系统等存储形态不同，它提供 RESTful API数据读写接口及丰富的 SDK 接口，并且常以网络服务的形式提供数据的访问。
>
> 对象存储不支持随机读写，只能进行全读和全写操作（无法直接在存储上进行数据更改，只能以文件为单位进行获取和删除），如果要修改数据，需要先在本地编辑完成后整文件上传。

> 亚马逊的对象存储服务叫 S3(simple storage service),阿里云的对象存储叫OSS(Object Storage Service)。

分布式存储

> 在海量数据和文件的场景下，单一服务器所能连接的物理介质有限，能提供的存储空间和IO性能也是有限的，所以可以通过多台服务器协同工作，每台服务器连接若干物理介质，再配合分布式存储系统和虚拟化技术，将底层多个物理硬件设备组合成一个统一的存储服务。一个分布式存储系统，可以同时提供块存储，文件存储和对象存储这三种存储方式。

1.3.2 NFS 基础

基础知识

简介

网络文件系统：Network File System (NFS)， 是由 SUN 公司研制的 UNIX 表示层协议，配合客户端和服务端软件，可以在本地挂载远程的存储空间，让本地用户和程序直接访问本地目录从而实现数据和存储空间的共享。

NFS 中的远程访问是基于 RPC (Remote Procedure Call Protocol，远程过程调用) 来实现的。

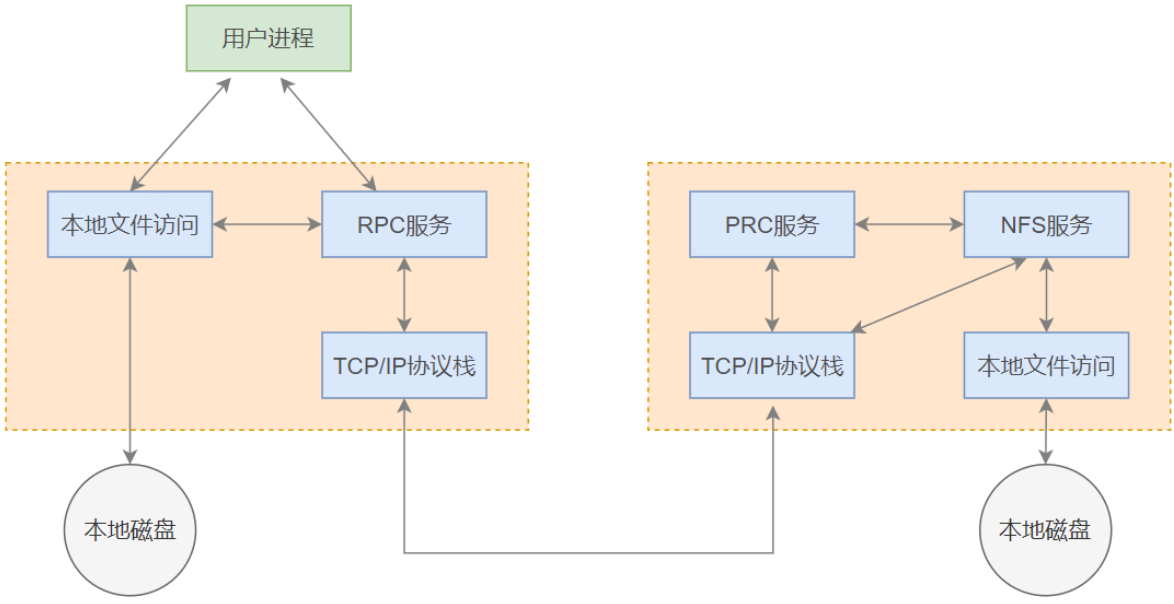
RPC采用C/S模式，客户机请求程序调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行。

性质特点

1 基于TCP/IP传输：
NFS通过网络文件系统协议，允许远程计算机像访问本地文件一样访问和操作远程文件，从而方便了多台计算机之间的文件共享和协作。
2 跨平台性：
NFS被广泛用于UNIX和Linux操作系统中，同时也可以支持在不同类型的系统之间通过网络进行文件共享，如FreeBSD、SCO、Solaris等异构操作系统平台。
3 简单易操作：
NFS提供了透明文件访问以及文件传输的功能，容易扩充新的资源或软件，而不需要改变现有的工作环境。
4 安全性：
NFS的安全性相对较低，因为它主要基于IP进行认证，且默认配置下对远程访问的用户（如root用户）会进行权限压缩（如映射为nfsnobody用户），以提高系统的安全性。但这也意味着需要额外的安全措施来确保数据的安全。

原理解读

工作原理



NFS 工作原理简图

NFS的工作原理基于客户端/服务器架构，由一个客户端程序和服务器程序组成。服务器程序向其他计算机提供对文件系统的访问，该过程称为“输出”。
NFS客户端程序对共享文件系统进行访问时，会将其从NFS服务器中“输送”出来。文件通常以块为单位进行传输，大小通常为8KB（尽管它可能会将操作分成更小尺寸的分片）。

在NFS的工作流程中，
前提：
- RPC服务需要先于NFS启动，NFS启动后，就会产生一些随机端口，然后向RPC服务去注册这些端口，
- RPC提供了一组与机器、操作系统以及底层传送协议无关的存取远程文件的操作，使得NFS能够在不同的计算机系统和设备之间共享文件和资源。

请求建立连接
- 客户端首先通过RPC（Remote Procedure Call，远程过程调用）协议到服务器端的RPC上要功能对应的端口号，然后客户端将请求发送给服务器所对应的端口号，并接收服务器返回的响应数据。
- 此后的数据传输便不再经过RPC，只在客户端和服务端之间进行数据传输。

应用场景

多个机器共享一台CDROM或其他设备：
通过NFS，可以将CDROM或其他设备挂载到NFS服务器上，然后通过网络共享给多个客户端使用。

用户home目录的集中管理：
在大型网络中，可以配置一台中心NFS服务器用来放置所有用户的home目录。这些目录能被输出到网络，以便用户不管在哪台工作站上登录，总能得到相同的home目录。

影视文件的共享：
不同客户端可以在NFS上观看影视文件，从而节省本地存储空间。

工作数据的备份：
在客户端完成的工作数据，可以备份保存到NFS服务器上用户自己的路径下，以确保数据的安全性和可恢复性。

1.3.3 NFS 部署

1.3.3.1 软件部署

环境部署

NFS 基于 C/S 模式实现，所以有客户端软件和服务端软件

```
在rocky 中安装，nfs-utils包含了客户端工具和服务端工具
[root@rocky9 ~]# yum install rpcbind nfs-utils
```

```
在ubuntu中安装服务端包
root@ubuntu24:~# apt install nfs-kernel-server

在ubuntu中安装客户端包
root@ubuntu24:~# apt install nfs-common
```

服务端依赖包，注册中心，nfs 服务中有大量的组件，部份组件端口并不固定，需要依赖rpcbind发布端口

查看软件包

```
root@ubuntu24:~# dpkg -l rpcbind
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name           Version       Architecture Description
+++-==============-=============-============-
=================================================
ii  rpcbind        1.2.6-2build1 amd64        converts RPC program numbers into
universal addresses
```

Rocky系统

```
[root@rocky9 ~]# rpm -q rpcbind
rpcbind-1.2.6-7.el9.x86_64
[root@rocky9 ~]# rpm -q nfs-utils
nfs-utils-2.5.4-26.el9_4.x86_64
```

查看服务状态

查看服务端端口

```
root@ubuntu24:~# ss -tunlp | grep rpc
udp UNCONN 0   0       127.0.0.1:603  0.0.0.0:*  users:
(("rpc.statd",pid=14843,fd=5))
udp UNCONN 0   0       0.0.0.0:38632  0.0.0.0:*  users:
(("rpc.statd",pid=14843,fd=8))
udp UNCONN 0   0       0.0.0.0:58332  0.0.0.0:*  users:
(("rpc.mountd",pid=15464,fd=4))
udp   UNCONN 0      0            0.0.0.0:111        0.0.0.0:*    users:
(("rpcbind",pid=6878,fd=5),("systemd",pid=1,fd=377))
udp   UNCONN 0      0               [::]:111           [::]:*    users:
(("rpcbind",pid=6878,fd=7),("systemd",pid=1,fd=379))
tcp   LISTEN 0      4096         0.0.0.0:111        0.0.0.0:*    users:
(("rpcbind",pid=6878,fd=4),("systemd",pid=1,fd=376))
tcp   LISTEN 0      4096            [::]:111           [::]:*    users:
(("rpcbind",pid=6878,fd=6),("systemd",pid=1,fd=378))
...
```

进程解读

rpcbind（也称为portmapper）是RPC服务的核心组件，它监听TCP和UDP的111端口，用于将RPC程序编号转换为网络地址和端口号。当客户端想要调用远程RPC服务时，它会首先联系rpcbind来获取服务的实际位置。在您的系统中，rpcbind正在正常运行，监听111端口。

rpc.statd是NFS（网络文件系统）状态监视守护进程的一部分，它用于跟踪NFS客户端和服务器之间的挂载状态，并维护锁和状态信息。虽然它通常不直接监听外部网络（在这里监听的是127.0.0.1，即本地回环地址），但它对于NFS服务的正常运行至关重要。

rpc.mountd是NFS挂载守护进程，它处理来自NFS客户端的挂载请求。rpc.mountd似乎正在监听随机的、高编号的UDP端口58332。这是正常的，因为NFS的rpc.mountd服务可以配置为监听动态分配的端口，以增加安全性。客户端在尝试挂载NFS文件系统时，会首先联系rpcbind来获取rpc.mountd的实际端口号。

每次重启服务，nfs的服务端，端口都在发生改变

```
重启服务
root@ubuntu24:~# systemctl restart nfs-server.service

再次查看端口，端口发生了变化
root@ubuntu24:~# ss -tunlp | grep rpc
udp UNCONN 0  0       0.0.0.0:36333  0.0.0.0:*  users:
(("rpc.mountd",pid=15507,fd=12))
udp UNCONN 0  0     127.0.0.1:603    0.0.0.0:*  users:
(("rpc.statd",pid=14843,fd=5))
udp UNCONN 0  0       0.0.0.0:38632  0.0.0.0:*  users:
(("rpc.statd",pid=14843,fd=8))
...
结果显示：
    由于服务端监听的端口会发生变化，所以需要依赖rpcbind对客户端提供注册服务，rpcbind端口不会发生变化
```

```
查看服务端使用的端口
root@ubuntu24:~# rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  portmapper   #rpcbind，早期叫portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    ...
```

1.3.3.2 rpcinfo

命令简介

```
    rpcinfo 可以访问指定的 RPC 服务器，显示其响应信息，从而查询出在该服务器上注册的RPC服务,如果不指定主机，则默认是当前主机。
```

格式解读

```
命令格式
    rpcinfo [-opt...] [host]

常用选项
    -p [host]           #查看注册在指定主机上的rpcbind V2 版本的 RPC 服务
    -m|-s [host]        #查看rpcbind的RPC服务，-s 简短格式显示
```

简单实践

```
显示已注册到本机的所有RPC服务
root@ubuntu24:~# rpcinfo

显示已注册到本机的rpcbind V2 版本的 RPC 服务
root@ubuntu24:~# rpcinfo -p
   program vers proto   port  service
    100000    4   tcp    111  portmapper
    100000    3   tcp    111  portmapper
    100000    2   tcp    111  portmapper
    ...
```

```
简短格式显示
root@ubuntu24:~# rpcinfo -s

查看指定主机
root@ubuntu24:~# rpcinfo -s 10.0.0.13
   program version(s) netid(s)                         service     owner
    100000  2,3,4     local,udp,tcp,udp6,tcp6          portmapper  superuser
    100024  1         tcp6,udp6,tcp,udp                status      125
    100005  3,2,1     tcp6,udp6,tcp,udp                mountd      superuser
    100003  4,3       tcp6,tcp                         nfs         superuser
    100227  3         tcp6,tcp                         nfs_acl     superuser
    100021  4,3,1     tcp6,udp6,tcp,udp                nlockmgr    superuser
```

1.3.3.3 exportfs

简介

```
exportfs 命令用于管理本机 NFS 文件系统，默认配置文件是 /etc/exports
```

命令解读

```
命令格式
    exportfs [-adfhioruvs] [host:/path]

常用选项
    -a          #全部挂载或者全部卸载
    -r          #重新挂载
    -u          #卸载
    -v          #显示本机共享
```

简单实践

```
修改配置文件后，加载配置
root@ubuntu24:~# exportfs -r

查看加载配置的效果
root@ubuntu24:~# exportfs -v

卸载操作
root@ubuntu24:~# exportfs -au
```

1.3.3.4 showmount

简介

```
shoumount 可以查看远程主机的共享设置
```

命令解读

```
命令格式
    showmount [ -opt... ] [ host ]

常用选项
    -h|--help               #显示帮助
    -a|--all                #显示已连接的客户端
    -e|--exports            #显示指定NFS服务器上的配置列表
```

简单实践

```
客户端，查看远程共享服务主机
root@ubuntu24:~# showmount -e 10.0.0.13
Export list for 10.0.0.13:
/data/dira 10.0.0.0/24
```

1.3.3.5 mount.nfs

简介

```
    客户端 NFS 服务挂载命令，将远程NFS 服务器上共享出来的目录挂载到本机，可以直接写成 mount
```

命令解读

```
命令格式：
    mount.nfs remotetarget dir [-rvVwfnsh] [-o nfsoptions]

常用选项
    -r                  #只读挂载
    -v                  #显示详细信息
    -V                  #显示版本
    -w                  #读写挂载
    -n                  #不更新/etc/fstab 文件，默认项
    -o                  #指定挂载参数
```

常用挂载参数

```
    fg                  #前台挂载，默认
    bg                  #后台挂载
    hard                #持续请求，如果挂不上，一直重试
    soft                #非持续挂载
    intr                #是否可以强制中断，配合hard 选项，如果挂不上，可以ctrl+c 中断
    rsize=n             #指定一次读的最大字节数，值必须为1024的倍数，最大为1048576，最小为
1024，
                        # 如果小于1024会被替换成4096，不指定由客户端和服务端协商
    wsize=n             #指定一次写的最大字节数，值必须为1024的倍数，最大 1048576，最小
1024，
                        # 如果小于 1024 会被替换成 4096，不指定则由客户端和服务端协商
    _netdev             #无网络服务时不挂载NFS资源
    vers=n|nfsvers=n    #指定nfs 协议版本号，如果NFS 服务端不支持此版本，则无法挂载，默认
4.2
```

简单实践

```
挂载nfs目录
root@ubuntu24:~# mount -o rw,fg,hard,intr 10.0.0.13:/testdir /mnt/nfs/
```

1.3.4 配置解读

1.3.4.1 配置基础

配置介绍

服务端主要进程

```
/usr/sbin/rpc.mountd        #挂载和卸载nfs文件系统，包括权限管理
/usr/sbin/rpc.nfsd          #主要的nfs进程，管理客户端是否可登录
/usr/sbin/rpc.statd         #非必要，检查文件一致性，可修复文件
```

日志目录

```
root@ubuntu24:~# tree /var/lib/nfs/
/var/lib/nfs/
├── etab
├── export-lock
├── nfsdcld
│   └── main.sqlite
├── rmtab
├── sm
├── sm.bak
├── state
└── v4recovery

4 directories, 5 files
```

共享规则配置

```
配置文件内容
root@ubuntu24:~# cat /etc/exports
...
# 共享目录路径        客户端访问时候获取的权限,多个权限可以通过空格隔开
# /srv/homes        hostname1(rw,sync,no_subtree_check) hostname2(...)
...

客户端主机写法
    anonymous,* 表示不限制，即任何主机都能访问
    单个主机可以写具体的IPV4,IPV6，FQDN，主机名
    网段 10.0.0.0/24 10.0.0.0/225.225.225.0 这两种格式都行
    主机名，主机名可以用通配符表示多台主机
    网域或主机组 @group_name
    gss/krb5i 这种写法来限制对使用rpcsec_gss安全性的客户端的访问

配置项写法
    默认是 (ro,sync,root_squash,no_all_squash)
```

```
主要选项
    ro                  只读
    rw                  读写
```

```
    root_squash          远程客户端主机上的 root 用户映射成 NFS 主机上的 UID=65534 的用户

    all_squash           所有客户端用户都映射成 NFS 主机上的 UID=65534 的用户
                             - 此项会覆盖 no_root_squash
    sync                 同步落盘，在请求数据时立即写入NFS的磁盘上，性能低，安全性高
    async                异步落盘，数据先暂存于缓冲区，再一次性落盘，性能高，安全性低
    no_subtree_check     不检查父目录权限
其他选项
    no_root_squash       远程客户端主机 root 用户映射成 NFS 服务器的 root
    anonuid              指定用户映射成特定的用户的UID
    anongid              指定用户映射成特定的用户的GID
    insecure             允许非授权访问
    subtree_check        如果共享子目录，强制NFS 检查父目录权限
    wdelay               多个用户写时等待同时写
    no_wdelay            多个用户写时不等待，立即写，当使用 async 项时，无需此设置
    hide                 不共享NFS 服务中的子目录，在 NFSv4中无效
    no_hide              共享NFS 服务器上的子目录，在 NFSv4中无效
    secure               NFS 服务通过1024以下的安全TCP/IP端口通讯
    insecure             NFS 服务通过1024以上的安全TCP/IP端口通讯
```

```
root@ubuntu24:~# id 65534
uid=65534(nobody) gid=65534(nogroup) 组=65534(nogroup)
```

配置示例

```
允许所有主机都可以访问共享目录，但是没有写权限
/path/to/dir   *

允许A主机进行读写操作，运行b主机读操作，其他主机不让访问
/path/to/dir   10.0.0.1(rw)   10.0.0.2

指定网段主机可以对共享目录进行读写操作
/path/to/dir   10.0.0.0/24(rw)
```

文件系统权限问题：

```
    共享目录在服务器上的文件系统权限可能不允许当前用户创建文件。例如，如果共享目录属于root用户，并且没有为"其他用户"设置适当的写权限，那么即使NFS权限设置为rw，非root用户也可能无法在该目录中创建文件。
    此时，可以通过修改共享目录的文件系统权限来解决此问题。具体来说，可以使用chown命令将共享目录的所有权更改为适当的用户或组，或者使用chmod命令调整目录的权限。
```

永久挂载

```
编辑/etc/fstab,挂载的文件系统类型是nfs
nfs_addr:/data/dira /local/dir       nfs    defaults,_netdev    0  0
```

1.3.4.2 共享实践

默认共享实践

```
准备共享目录
root@ubuntu24:~# mkdir /data/dir{a,b}
root@ubuntu24:~# cp /etc/fstab /data/dira/
```

```
定制专属的配置
root@ubuntu24:~# cat /etc/exports
/data/dira  *                          # 所有主机都可以挂载，权限是默认的权限
```

配置加载动作

```
读取配置
root@ubuntu24:~# exportfs -r
或
root@ubuntu24:~# systemctl restart nfs-server.service
```

测试效果

```
自己查看信息
root@ubuntu24:~# exportfs -v
/data/dira      <world>
(sync,wdelay,hide,no_subtree_check,sec=sys,ro,secure,root_squash,no_all_squash)
默认权限解读：
    sync：表示所有数据在请求时同步写入共享文件系统，在数据被写入磁盘之前，文件系统将等待写入完成。
    wdelay：这是写入延迟的一个设置，当多个用户尝试写入时，它们的写操作可能会被延迟并归组到一起执行。
    hide：在NFS共享目录中不共享其子目录。如果选择此选项，则共享目录的子目录不会被共享出去。
    no_subtree_check：不检查父目录权限。在访问共享目录时，NFS服务器不会检查从根目录到当前目录路径上的所有挂载点，这可以提高访问速度，但在某些情况下可能会导致安全问题。
    sec=sys：指定NFS使用的安全机制。sys通常依赖于主机名和用户ID来验证访问权限。
    ro：只读权限。如果设置了这个选项，客户端将只能读取共享目录中的文件，而不能写入。
    secure：表示NFS通过1024以下的安全TCP/IP端口发送数据。但可能会限制一些非标准端口的访问。
    root_squash：root用户的所有请求映射成如anonymous用户一样的权限（默认）。从而保护NFS服务器的安全。
    no_all_squash：保留共享文件的UID和GID（默认与root_squash相对）。
```

```
在远程主机上测试
root@ubuntu24:~# showmount -e 10.0.0.13
Export list for 10.0.0.13:
/data/dira *
```

挂载实践

```
客户端挂载实践
root@ubuntu24:~# mkdir -pv /data/dir{1,2}
mkdir: created directory '/data'
mkdir: created directory '/data/dir1'
mkdir: created directory '/data/dir2'

将远程主机上的 /data/dira 目录挂载到本机的 /data/dir1 上
root@ubuntu24:~# mount 10.0.0.13:/data/dira /data/dir1

查看挂载效果
root@ubuntu24:~# mount
10.0.0.13:/data/dira on /data/dir1 type nfs4
(ro,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,hard,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=10.0.0.16,local_lock=none,addr=10.0.0.13)
```

查看挂载效果

```
确认目录
root@ubuntu24:~# df -h /data/dir1/
Filesystem             Size  Used Avail Use% Mounted on
10.0.0.208:/data/dira   97G  7.4G   85G   9% /data/dir1
root@ubuntu24:~# ls -l /data/dir1
total 4
-rw-r--r-- 1 root root 657 Jul  9 09:58 fstab
```

确认挂载权限

```
可读
root@ubuntu24:~# cat /data/dir1/fstab
# /etc/fstab: static file system information.
...

不可写
root@ubuntu24:~# echo "123" >> /data/dir1/fstab
-bash: /data/dir1/fstab: Read-only file system
root@ubuntu24:~# touch /data/dir1/test
touch: cannot touch '/data/dir1/test': Read-only file system
```

1.3.4.3 挂载权限实践

指定客户端主机权限设置

```
修改默认配置
root@ubuntu24:~# cat /etc/exports
#10.0.0.16 可读写，  10.0.0.12 默认选项
/data/dira  10.0.0.16(rw)  10.0.0.12
```

配置生效

```
生效
root@ubuntu24:~# exportfs -r

查看效果
root@ubuntu24:~# exportfs -v
/data/dira
10.0.0.16(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,root_squash,no_all
_squash)
/data/dira
10.0.0.12(sync,wdelay,hide,no_subtree_check,sec=sys,ro,secure,root_squash,no_all
_squash)
```

挂载操作测试

```
16 主机上测试，还是可读不可写
root@ubuntu24-16:~# ls -l /data/dir1/
total 4
-rw-r--r-- 1 root root 657 Jul  9 09:58 fstab
root@ubuntu24-16:~# cat /data/dir1/fstab
# /etc/fstab: static file system information.

测试创建文件，依然报错
root@ubuntu24-16:~# touch /data/dir1/test-16
touch: cannot touch '/data/dir1/test-16': Permission denied
```

为什么不可写?

```
因为对于共享目录来说，它对于其他用户不具有写的功能
root@ubuntu24:~# ll /data/dira -d
drwxr-xr-x 2 root root 4096 10月 18 20:23 /data/dira/
```

```
为其他用户添加写权限
root@ubuntu24:~# chmod o+w /data/dira
root@ubuntu24:~# ll /data/dira -d
drwxr-xrwx 2 root root 4096 10月 18 20:23 /data/dira/
```

```
16主机尝试添加文件
root@ubuntu24-16:~# ls /data/dir1/test-16
/data/dir1/test-16
root@ubuntu24-16:~# ls /data/dir1/ -l
总计 4
-rw-r--r-- 1 root   root    473 11月 26 23:12 fstab      # 显示原来的文件权限身份
-rw-r--r-- 1 nobody nogroup   0 11月 26 23:30 test-16    # 是默认的nobody身份
结果显示：
    权限赋予操作成功了
```

授权的12主机，进行挂载测试

```
12 主机上测试
[root@rocky9 ~]# mkdir /data/dir{1,2}

检查效果
[root@rocky9 ~]# showmount  -e 10.0.0.13
Export list for 10.0.0.13:
/data/dira 10.0.0.12,10.0.0.16

挂载
[root@rocky9 ~]# mount 10.0.0.13:/data/dira /data/dir1
[root@rocky9 ~]# df /data/dir1/
文件系统                 1K-块      已用       可用 已用% 挂载点
10.0.0.13:/data/dira 205310976 10975232 183833600    6% /data/dir1
```

```
验证权限
[root@rocky9 ~]# ls /data/dir1
fstab  test-16
[root@rocky9 ~]# touch /data/dir1/test-12
touch: 无法创建 '/data/dir1/test-12': 只读文件系统
结果显示：
    可看不可写
```

使用授权地址范围之外的主机进行测试效果

```
非授权用户查看效果
root@ubuntu24:~# showmount -e 10.0.0.13
Export list for 10.0.0.13:
/data/dira 10.0.0.12,10.0.0.16
root@ubuntu24:~# mkdir /data/0.13

结果显示，无法挂载
root@ubuntu24:~# mount 10.0.0.13:/data/dira /data/0.13
mount.nfs: access denied by server while mounting 10.0.0.13:/data/dira
结果显示：
    无法正常的挂载操作
```

1.3.4.4 限制身份显示

现状演示

现状

```
    根据我们之前的演示，对于root的身份操作文件，全部映射为 nobody的用户，而对于普通用户的身份操作文件，是没有做任何身份映射的。
```

演示效果

```
16主机使用普通用户操作
root@ubuntu24-16:~# useradd -m zhangsan
root@ubuntu24-16:~# su - zhangsan
$ echo zhangsan > /data/dir1/zhangsan-1
$ exit

root@ubuntu24-16:~# su - sswang
sswang@ubuntu24-16:~$ echo sswang > /data/dir1/sswang-1
sswang@ubuntu24-16:~$ ll /data/dir1/
...
-rw-r--r-- 1 root   root     473 11月 26 23:12 fstab
-rw-rw-r-- 1 sswang sswang     7 11月 26 23:41 sswang-1      # 普通用户的文件权限保留了
-rw-r--r-- 1 nobody nogroup    0 11月 26 23:30 test-16
-rw-rw-r-- 1 zhangsan zhangsan    9 11月 26 23:45 zhangsan-1 # 普通用户的文件权限保留了
```

服务器端查看效果

```
root@ubuntu24:~# ll /data/dira/
总计 20
drwxr-xrwx 2 root   root    4096 11月 26 23:45 ./
drwxr-xr-x 7 root   root    4096 11月 26 23:33 ../
-rw-r--r-- 1 root   root     473 11月 26 23:12 fstab
-rw-rw-r-- 1 sswang sswang     7 11月 26 23:41 sswang-1
-rw-r--r-- 1 nobody nogroup    0 11月 26 23:30 test-16
-rw-rw-r-- 1   1001    1001    9 11月 26 23:45 zhangsan-1
结果显示：
    因为服务端，没有id为1001用户，所以直接显示成对应的id值
```

服务端增加一个id用户

```
增加一个用户id为1001的普通用户
root@ubuntu24:~# useradd -s /bin/bash -m -u 1001 jerry

再次查看文件权限效果
root@ubuntu24:~# ll /data/dira/
总计 20
drwxr-xrwx 2 root   root    4096 11月 26 23:45 ./
drwxr-xr-x 7 root   root    4096 11月 26 23:33 ../
-rw-r--r-- 1 root   root     473 11月 26 23:12 fstab
-rw-rw-r-- 1 sswang sswang     7 11月 26 23:41 sswang-1
-rw-r--r-- 1 nobody nogroup    0 11月 26 23:30 test-16
-rw-rw-r-- 1 jerry  jerry      9 11月 26 23:45 zhangsan-1
结果显示：
    该文件显示为了 服务端的 用户id为1001 的 jerry了
```

注意：

```
    通过 普通用户身份 不映射的 特点，我们可以实现，多web主机上，同时挂载一个文件目录，然后在所有的主机上，同时创建 同一个 id的 同名 www 用户，这样，就可以实现 "统一用户映射" 的效果了。
```

隐藏身份演示

修改共享文件的权限效果

```
修改默认配置
root@ubuntu24:~# cat /etc/exports
#10.0.0.16 可读写，  10.0.0.12 默认选项
/data/dira  10.0.0.16(rw,no_root_squash,all_squash)  10.0.0.12
注意：
    all_squash 会覆盖no_root_squash的功能
    多个选项值之间，不允许出现空格，必须以逗号紧密连接在一起
```

配置生效

```
生效
root@ubuntu24:~# exportfs -r

查看效果
root@ubuntu24:~# exportfs -v
/data/dira
 10.0.0.16(sync,wdelay,hide,no_subtree_check,sec=sys,rw,secure,no_root_squash,al
l_squash)
/data/dira
 10.0.0.12(sync,wdelay,hide,no_subtree_check,sec=sys,ro,secure,root_squash,no_al
l_squash)
```

16主机，进行root和普通用户的实践

```
root@ubuntu24-16:~# userdel lisi
root@ubuntu24-16:~# useradd -m -s /bin/bash lisi
root@ubuntu24-16:~# su - lisi
lisi@ubuntu24-16:~$ echo lisi > /data/dir1/lisi-1
lisi@ubuntu24-16:~$ exit
注销
```

```
root@ubuntu24-16:~# su - sswang
sswang@ubuntu24-16:~$ echo sswang > /data/dir1/sswang-2
sswang@ubuntu24-16:~$ exit
注销
```

确认效果

```
root@ubuntu24-16:~# echo root > /data/dir1/root-1
root@ubuntu24-16:~# ll /data/dir1/
总计 32
drwxr-xrwx 2 root     root     4096 11月 27 00:00 ./
drwxr-xr-x 4 root     root     4096 10月 18 20:09 ../
-rw-r--r-- 1 root     root      473 11月 26 23:12 fstab
-rw-rw-r-- 1 nobody   nogroup     5 11月 26 23:59 lisi-1
-rw-r--r-- 1 nobody   nogroup     5 11月 27 00:00 root-1
-rw-rw-r-- 1 sswang   sswang      7 11月 26 23:41 sswang-1
-rw-rw-r-- 1 nobody   nogroup     7 11月 26 23:59 sswang-2
-rw-r--r-- 1 nobody   nogroup     0 11月 26 23:30 test-16
-rw-rw-r-- 1 zhangsan zhangsan    9 11月 26 23:45 zhangsan-1
结果显示：
    所有新创建的用户身份都变成了 nobody
```

1.4 实时同步

1.4.1 基础知识

1.4.1.1 数据同步介绍

简介

场景需求

```
    在生产环境中，某些场景下，要将数据或文件进行实时同步，保证数据更新后其它节点能立即获得最新的数据。
```

数据同步方式

```
PULL：拉，
    使用定时任务的方式配合同步命令或脚本等，从指定服务器上将数据同步到本地，一般是周期性定时同步

PUSH：推，
    如果当前机器上的数据或文件发生更新了，立即推送到指定的节点上，可以做到实时同步
```

方案

实时同步解决方案

```
rsync +inotify:
    rsync 是一个常用的文件同步工具，而 inotify 是一个 Linux 内核功能，可以监控文件系统事件。你可以结合这两个工具来实现实时同步。

Lsyncd
    Lsyncd 是一个基于 rsync 和 inotify 的实时文件同步工具，它简化了同步配置的复杂性。

Sersync
    类似 于inotify，同样用于监控，但它克服了 inotify 的缺点，inotify 最大的不足是会产生重复事件。Sersync使用多线程进行同步，还自带 crontab 功能。
```

```
Samba/CIFS 挂载
    如果你需要将一个目录实时挂载到另一个目录，并且你的网络环境允许，你可以使用 Samba/CIFS 挂载来实现。

NFS
    NFS（Network File System）也是一种常见的网络文件系统，适用于 Linux 系统之间的文件共享和同步。
```

1.4.1.2 inotify + rsync 实时同步

基础知识

实现原理

```
    利用内核中的 inotify 监控指定目录，当目录中的文件或数据发生变化时，立即调用 rsync 服务将数据推送到远程主机上。
```

inotify 服务

```
    inotify 是一个内核用于通知用户空间程序文件系统变化的机制，在监听到文件系统发生变化后，会向相应的应用程序发送事件，如文件增加，修改，删除等事件发生后可以立即让用户空间知道。
```

```
https://github.com/rvoicilas/inotify-tools          #项目地址
```

inotify 是内核中的功能，在2.6.13版本的内核开始都默认存在。

```
Ubuntu系统
root@ubuntu24-13:~# uname -r
6.8.0-48-generic
root@ubuntu24-13:~# grep -i inotify /boot/config-6.8.0-48-generic
CONFIG_INOTIFY_USER=y
```

```
ubuntu系统
[root@rocky9-12 ~]# uname -r
5.14.0-427.13.1.el9_4.x86_64
[root@rocky9-12 ~]# grep -i notify /boot/config-5.14.0-427.13.1.el9_4.x86_64
CONFIG_FB_NOTIFY=y
CONFIG_DMABUF_MOVE_NOTIFY=y
CONFIG_FSNOTIFY=y
CONFIG_DNOTIFY=y
CONFIG_INOTIFY_USER=y
CONFIG_FANOTIFY=y
CONFIG_FANOTIFY_ACCESS_PERMISSIONS=y
```

内核参数

inotify 内核参数

```
查看inotify的内核参数文件
root@ubuntu24-13:~# ls -l /proc/sys/fs/inotify
总计 0
-rw-r--r-- 1 root root 0 11月 26 16:01 max_queued_events
-rw-r--r-- 1 root root 0 11月 26 16:01 max_user_instances
-rw-r--r-- 1 root root 0 11月 26 15:58 max_user_watches
```

```
root@ubuntu24-13:~# cat /proc/sys/fs/inotify/max_queued_events
16384
参数解读：
    inotify 事件队列最大长度，如值太小会出现 Event Queue Overflow 错误，默认 16384，
    生产环境建议调大,比如 327679

root@ubuntu24:~# cat /proc/sys/fs/inotify/max_user_instances
128
参数解读：
    每个用户创建inotify实例最大值，默认值 128

root@ubuntu24-13:~# cat /proc/sys/fs/inotify/max_user_watches
65536
参数解读：
    每个inotify实例可以监视的文件的总数量
    该值在rocky系统中，默认值是 12820
```

```
也就是说：
    一个用户可以最多开128个inotify实例，每个实例最多可以监控65536个文件
```

内核参数如果需要修改，可以配置文件

```
定制专属参数配置文件
root@ubuntu24:~# cat /etc/sysctl.d/inotify.conf
fs.inotify.max_queued_events=66666
fs.inotify.max_user_instances=256
fs.inotify.max_user_watches=100000

应用内核参数文件
root@ubuntu24:~# sysctl -p /etc/sysctl.d/inotify.conf
fs.inotify.max_queued_events = 66666
fs.inotify.max_user_instances = 256
fs.inotify.max_user_watches = 100000
```

1.4.2 环境部署

1.4.2.1 软件部署

软件简介

安装软件

```
    inotify 是内核中的功能模块，只能通过调用 API 接口的形式使用其功能，我们可以通过相关软件来对其进行操作，能实现内核中 inotify 调用的软件主要有以下几个：
    inotify-tools，sersync，lrsyncd
```

环境部署

安装软件

```
centos系统
[root@rocky9-12 ~]# yum install epel-release
[root@rocky9-12 ~]# yum install inotify-tools

查看文件
[root@rocky9-12 ~]# rpm -ql inotify-tools
/usr/bin/inotifywait
/usr/bin/inotifywatch
...
```

```
ubuntu系统
root@ubuntu24-13:~# apt install inotify-tools

查看软件
root@ubuntu24-13:~# dpkg -L inotify-tools
...
/usr/bin/fsnotifywait
/usr/bin/fsnotifywatch
/usr/bin/inotifywait
/usr/bin/inotifywatch
```

主要工具解读

```
/usr/bin/fsnotifywait       #fsnotify监控工具，fsnotify 是 inotify 的新版本
/usr/bin/fsnotifywatch      #fsnotify统计工具
/usr/bin/inotifywait        #实时监控指定目录的所有事件，在被监控的文件或目录上等待特定事件发生(open,close,write..)
/usr/bin/inotifywatch       #收集被监控的文件系统使用的统计数据，指文件系统事件发生的次数统计
```

1.4.2.2 命令解读

命令解读

命令格式

```
    inotifywait [ options ] file1 [ file2 ] [ file3 ] [ ... ]
```

常用选项

```
    -h|--help                   #显示帮助
    @<file>                     #排除指定文件
    --exclude <pattern>         #不监控指定的文件或目录，正则匹配
    --excludei <pattern>        #不监控指定的文件或目录，正则匹配，不区分大小写
    --include <pattern>         #仅监控指定的文件或目录，正则匹配
    --includei <pattern>        #仅监控指定的文件或目录，正则匹配，不区分大小写
    -m|--monitor                #始终保持监听，直到超时
    -d|--daemon                 #以守护进程方式执行，配合-o使用
    -P|--no-dereference         #不跟随软链接
    -r|--recursive              #对目录递归监控，即监控目录内所有内容
    --fromfile <file>           #从文件中读取要监控的内容
    -o|--outfile <file>         #将执行结果输出到指定文件
    -s|--syslog                 #错误输出到syslog，而不是标准错误输出
    -q|--quiet                  #简短输出，仅输出事件信息
    -qq                         #禁用输出
    --format <fmt>              #指定输出格式
    --no-newline                #不换行输出，配合--format字段
    --timefmt <fmt>             #指定时间输出格式
    -c|--csv                    #以CSV格式输出
    -t|--timeout <seconds>      #指定超时时长，值为0则永远不超时
    -e|--event <event1>         #只监听指定事件，默认监听所有事件
```

--format 选项可用字段

```
    %T                          #输出时间格式中定义的时间格式信息，
                                #通过 --timefmt option 语法格式指定时间信息
    %w                          #事件出现时，监控文件或目录的名称信息，相当于dirname
    %f                          #事件出现时，将显示监控目录下触发事件的文件或目录信息，
                                #否则为空，相当于basename
    %e                          #显示发生的事件信息，不同的事件默认用逗号分隔
    %Xe                         #显示发生的事件信息，不同的事件指定用X进行分隔
```

--timefmt 选项可用字段

```
    %Y                          #年份信息，包含世纪信息
    %y                          #年份信息，不包括世纪信息
    %m                          #显示月份，范围 01-12
    %d                          #每月的第几天，范围是 01-31
    %H                          #小时信息，使用 24小时制，范围 00-23
    %M                          #分钟，范围 00-59
    %S                          #秒，范例 0-60
```

-e 选项可以指定的事件

```
    access                      #文件或目录内容被读取
    modify                      #内容发生变化
    attrib                      #属性发生变化
    close_write                 #写入模式打开后关闭
    close_nowrite               #只读打开后关闭
    close                       #关闭事件，无关模式
    open                        #打开事件，无关模式
    moved_to                    #文件或目录被移动到监控的目录中
    moved_from                  #文件或目录从监控的目录中被移动
    move                        #文件或目录不管移动到或是移出监控目录都触发事件
    move_self                   #被移动之后同名文件不再被监控
    create                      #创建文件或目录
    delete                      #文件或目录被删除
    delete_self                 #被删除之后同名文件不再被监控
    unmount                     #取消挂载
```

1.4.2.3 简单监控

实践1：只监控一个事件

准备工作

```
root@ubuntu24-13:~# mkdir /data/dir{1,2} -p
root@ubuntu24-13:~# tree /data/
/data/
├── dir1
└── dir2

3 directories, 0 files
```

开始监控

```
root@ubuntu24-13:~# inotifywait /data/
Setting up watches.          #表示 inotifywait 正在设置监控（watches）。这是初始化过程的一部分。
Watches established.         #表示监控已经成功建立。inotifywait 正在等待data目录中的文件系统事件。
# 处于阻塞状态 ...
```

在另一个终端B中执行如下命令

```
root@ubuntu24-13:~# ls /data/
dir1  dir2
```

回到监控的终端，查看效果

```
root@ubuntu24-13:~# inotifywait /data/
Setting up watches.
Watches established.
/data/ OPEN,ISDIR            # 这是 inotifywait 检测到的一个事件。
                             # 它表示 /data/ 目录被打开（OPEN），并且 ISDIR 标志表示这是
一个目录
                             #（而不是文件）。在Linux中，打开目录通常意味着某个进程正在读取
目录内容，
                             # 比如列出目录中的文件和子目录。
```

实践2：持续监控

在终端A中，再次监控data目录，使用 -m 实现持续监控

```
root@ubuntu24-13:~# inotifywait -m /data/
Setting up watches.
Watches established.
# 处于阻塞状态 ...
```

在另一个终端B中执行如下命令

```
root@ubuntu24-13:~# ls /data/
dir1  dir2
root@ubuntu24-13:~# ls /data/dir1
root@ubuntu24-13:~# ls /data/dir2
```

回到终端A查看效果

```
root@ubuntu24-13:~# inotifywait -m /data/
Setting up watches.
Watches established.
/data/ OPEN,ISDIR                       # 检测到 /data/ 目录被打开，
/data/ ACCESS,ISDIR                     # 检测到对 /data/ 目录的访问。
/data/ CLOSE_NOWRITE,CLOSE,ISDIR        # 检测到对 /data/ 目录的关闭操作
---- 这些事件序列表示 /data/dir1 目录被打开、访问、然后关闭（没有写入）。
/data/ OPEN,ISDIR dir1
/data/ ACCESS,ISDIR dir1
/data/ ACCESS,ISDIR dir1
/data/ CLOSE_NOWRITE,CLOSE,ISDIR dir1
---- 这些事件表示 /data/dir2 目录也经历了打开、访问、然后关闭（没有写入）的过程。
/data/ OPEN,ISDIR dir2
/data/ ACCESS,ISDIR dir2
/data/ ACCESS,ISDIR dir2
/data/ CLOSE_NOWRITE,CLOSE,ISDIR dir2

指令解读：
    ISDIR 表示这是一个目录打开事件。
    ACCESS 事件通常表示读取目录内容，但不一定意味着实际读取了文件。
    CLOSE 表示一般关闭事件
    CLOSE_NOWRITE 表示没有写入的关闭事件
```

1.4.2.4 递归监控

正常的监控演示-看不到文件创建删除效果

在终端B中，进行基本的文件操作

```
root@ubuntu24-13:~# ls /data/dir1/
root@ubuntu24-13:~# touch /data/dir1/test.txt
root@ubuntu24-13:~# rm -f /data/dir1/test.txt
```

回到终端A查看效果

```
root@ubuntu24-13:~# inotifywait -m /data/
...
/data/ OPEN,ISDIR dir1
/data/ ACCESS,ISDIR dir1
/data/ CLOSE_NOWRITE,CLOSE,ISDIR dir1
结果显示：
    没有监控到 对于文件的创建和删除的动作
```

递归监控演示-可以看到文件创建删除效果

在终端A关闭之前命令，通过 -r 实现递归的监控操作

```
root@ubuntu24-13:~# inotifywait -m -r /data/
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
# 处于阻塞状态 ...
```

在终端B中，进行基本的文件操作

```
root@ubuntu24-13:~# ls /data/dir1/
root@ubuntu24-13:~# touch /data/dir1/test.txt
root@ubuntu24-13:~# rm -f /data/dir1/test.txt
```

回到终端A查看效果

```
root@ubuntu24-13:~# inotifywait -m -r /data/
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
---- 查看二层目录的监视过程
/data/ OPEN,ISDIR dir1
/data/dir1/ OPEN,ISDIR
/data/ ACCESS,ISDIR dir1
/data/dir1/ ACCESS,ISDIR
/data/ ACCESS,ISDIR dir1
/data/dir1/ ACCESS,ISDIR
/data/ CLOSE_NOWRITE,CLOSE,ISDIR dir1
/data/dir1/ CLOSE_NOWRITE,CLOSE,ISDIR
/data/dir1/ CREATE test.txt         # 在 /data/dir1/ 目录中创建了一个名为 test.txt 的
文件。
/data/dir1/ OPEN test.txt           # test.txt 文件被打开。
/data/dir1/ ATTRIB test.txt         # test.txt 文件的属性被修改。
/data/dir1/ CLOSE_WRITE,CLOSE test.txt  # test.txt 文件在完成写入后被关闭。
/data/dir1/ DELETE test.txt         # test.txt 文件被删除。
```

```
属性解析：
    CREATE 事件表示文件或目录的创建。
    OPEN 事件表示文件或目录的打开。
    ATTRIB 事件表示文件或目录的属性变化，这可能包括权限、所有权、时间戳等的变化。
    CLOSE_WRITE 表示文件在关闭前进行了写入操作。
    CLOSE 表示文件或目录的关闭。
    CLOSE_WRITE,CLOSE 通常一起出现，表示文件在完成写入操作后被关闭。
    DELETE 事件表示文件或目录的删除。
```

1.4.2.4 监控输出和输入

实践1：将结果输出到文件

在终端A关闭之前命令，通过 -o 选项将结果输出到一个文件

```
root@ubuntu24-13:~# inotifywait -m -r /data/ -o inotify.txt
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
# 处于阻塞状态 ...
```

在终端B进行基本操作

```
root@ubuntu24-13:~# ls /data/dir1/
root@ubuntu24-13:~# touch /data/dir1/test.txt
root@ubuntu24-13:~# rm -f /data/dir1/test.txt
```

在终端A没有任何信息显示

```
root@ubuntu24-13:~# inotifywait -m -r /data/ -o inotify.txt
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
# 处于阻塞状态 ...
```

查看信息输出文件效果

```
root@ubuntu24-13:~# cat inotify.txt
/data/ OPEN,ISDIR dir1
/data/dir1/ OPEN,ISDIR
/data/ ACCESS,ISDIR dir1
/data/dir1/ ACCESS,ISDIR
/data/ ACCESS,ISDIR dir1
/data/dir1/ ACCESS,ISDIR
/data/ CLOSE_NOWRITE,CLOSE,ISDIR dir1
/data/dir1/ CLOSE_NOWRITE,CLOSE,ISDIR
/data/dir1/ CREATE test.txt
/data/dir1/ OPEN test.txt
/data/dir1/ ATTRIB test.txt
/data/dir1/ CLOSE_WRITE,CLOSE test.txt
/data/dir1/ DELETE test.txt
结果显示：
    各种操作的监控状态都采集到了
```

实践2：从文件中读取要监控的内容

准备工作，将需要监控的目录，放到文件中

```
root@ubuntu24-13:~# echo /data/ > monitor_file.txt
```

在A终端，通过 --fromfile 实现文件内容的监控

```
root@ubuntu24-13:~# inotifywait -rm --fromfile monitor_file.txt
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
# 处于阻塞状态 ...
```

终端B进行基本操作

```
root@ubuntu24-13:~# touch /data/dir1/test1.txt
root@ubuntu24-13:~# rm -f /data/dir1/test1.txt
root@ubuntu24-13:~# echo nihao > /data/dir1/test.txt
root@ubuntu24-13:~# rm -f /data/dir1/test1.txt
```

回到A终端，确认监控效果

```
root@ubuntu24-13:~# inotifywait -rm --fromfile monitor_file.txt
Setting up watches.  Beware: since -r was given, this may take a while!
Watches established.
/data/ OPEN,ISDIR dir1
/data/dir1/ OPEN,ISDIR
/data/ ACCESS,ISDIR dir1
/data/dir1/ ACCESS,ISDIR
/data/ ACCESS,ISDIR dir1
/data/dir1/ ACCESS,ISDIR
/data/ CLOSE_NOWRITE,CLOSE,ISDIR dir1
/data/dir1/ CLOSE_NOWRITE,CLOSE,ISDIR
/data/dir1/ CREATE test1.txt
/data/dir1/ OPEN test1.txt
/data/dir1/ ATTRIB test1.txt
/data/dir1/ CLOSE_WRITE,CLOSE test1.txt
/data/dir1/ DELETE test1.txt
/data/dir1/ OPEN test.txt
/data/dir1/ MODIFY test.txt
/data/dir1/ MODIFY test.txt
/data/dir1/ CLOSE_WRITE,CLOSE test.txt
```

1.4.2.5 监控格式

实践1：后台守护执行，并指定输出格式

需要提前指定信息输出的文件，否则会发生报错

```
root@ubuntu24-13:~# inotifywait -drq /data/ -o inotify.log --timefmt "%Y-%m-%d
%H:%M:%S" --format "%T %w %f event: %e"
No such file or directory: inotify.log
```

准备好文件后，进行监控

```
创建信息输出的文件
root@ubuntu24-13:~# touch inotify.log

开始监控
root@ubuntu24-13:~# inotifywait -drq /data/ -o inotify.log --timefmt "%Y-%m-%d
%H:%M:%S" --format "%T %w %f event: %e"

查看文件内容 -- 内容为空
root@ubuntu24-13:~# cat inotify.log
root@ubuntu24-13:~#
```

操作演示效果

```
终端B进行如下操作
root@ubuntu24-13:~# touch /data/dir1/test1.txt
root@ubuntu24-13:~# echo nihao > /data/dir1/test.txt
root@ubuntu24-13:~# rm -f /data/dir1/test1.txt
root@ubuntu24-13:~# rm -f /data/dir1/test.txt
```

```
终端A查看效果
root@ubuntu24-13:~# cat inotify.log
2024-11-26 16:55:38 /data/ dir1 event: OPEN,ISDIR
2024-11-26 16:55:38 /data/dir1/  event: OPEN,ISDIR
...
2024-11-26 16:55:42 /data/dir1/ test.txt event: CLOSE_WRITE,CLOSE
2024-11-26 16:55:44 /data/dir1/ test1.txt event: DELETE
2024-11-26 16:55:48 /data/dir1/ test.txt event: DELETE
结果显示：
    内容都是格式化方式输出了
```

实践2：指定输出格式，且只记录特定事件

在终端A进行定制化格式操作

```
root@ubuntu24-13:~# inotifywait -mrq /data/ --timefmt "%F %H:%M:%S" --format "%T
%w%f event:%;e" -e create,delete,moved_to,close_write,attrib
# 处于阻塞状态 ...
```

在终端B进行正常操作

```
root@ubuntu24-13:~# ls /data/dir1/
root@ubuntu24-13:~# echo nihao > /data/dir1/test.txt
root@ubuntu24-13:~# rm -f /data/dir1/test.txt
```

在终端A确认监控效果

```
root@ubuntu24-13:~# inotifywait -mrq /data/ --timefmt "%F %H:%M:%S" --format "%T
%w%f event:%;e" -e create,delete,moved_to,close_write,attrib
# 下面的信息，就是监控出来的结果
2024-11-26 16:59:46 /data/dir1/test.txt event:CREATE
2024-11-26 16:59:46 /data/dir1/test.txt event:CLOSE_WRITE;CLOSE
2024-11-26 16:59:54 /data/dir1/test.txt event:DELETE
```

1.4.3 rsync 服务

1.4.3.1 软件基础

rsync

简介

```
    rsync 常用于做为 linux系统下的数据镜像备份工具，实现远程同步，支持本地复制，或者与其他SSH、rsync主机同步数据，支持增量备份，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数据同步
```

```
http://rsync.samba.org/          #官方网站
```

软件安装

安装软件

```
Centos系统
[root@rocky9-12 ~]# yum install rsync

查看版本
[root@rocky9-12 ~]# rsync --version
rsync  version 3.2.3  protocol version 31
...
```

```
Ubuntu系统
root@ubuntu24-13:~# apt install rsync

查看版本
root@ubuntu24-13:~# rsync --version
rsync  version 3.2.7  protocol version 31
Copyright (C) 1996-2022 by Andrew Tridgell, Wayne Davison, and others.
Web site: https://rsync.samba.org/
Capabilities:
    64-bit files, 64-bit inums, 64-bit timestamps, 64-bit long ints,
    socketpairs, symlinks, symtimes, hardlinks, hardlink-specials,
    hardlink-symlinks, IPv6, atimes, batchfiles, inplace, append, ACLs,
    xattrs, optional secluded-args, iconv, prealloc, stop-at, no crtimes
Optimizations:
    SIMD-roll, no asm-roll, openssl-crypto, no asm-MD5
Checksum list:
    xxh128 xxh3 xxh64 (xxhash) md5 md4 sha1 none
Compress list:
    zstd lz4 zlibx zlib none
Daemon auth list:
    sha512 sha256 sha1 md5 md4

rsync comes with ABSOLUTELY NO WARRANTY.  This is free software, and you
are welcome to redistribute it under certain conditions.  See the GNU
General Public Licence for details.
```

1.4.3.2 命令解读

命令格式

rsync 有三种工作方式

```
1. 本地文件系统上实现同步。
2. 本地主机使用远程ssh和远程主机通信。
3. 本地主机通过网络套接字连接远程主机上的 rsync daemon。
```

```
    前两者的本质是通过本地或远程shell，
    第3种方式则是让远程主机上运行 rsyncd 服务，通过监听端口，等待客户端的连接，使用的rsync的专有协议。
```

常见命令样式

```
本地文件同步
rsync [OPTION]... SRC [SRC]... DEST
```

```
远程文件同步
rsync [OPTION]... [USER@]HOST:SRC [DEST]            #PULL 拉，将远程主机上的文件拉到本地
rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST     #PUSH 推，将本地文件推送到远程主机上
```

```
守护进程方式
rsync [OPTION]... [USER@]HOST::SRC [DEST]                       #将远程主机上的文件拉到本地
rsync [OPTION]... rsync://[USER@]HOST[:PORT]/SRC [DEST]         #将远程主机上的文件拉到本地

rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST                #将本地文件推送到远程主机上
rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST  #将本地文件推送到远程主机上
```

常用选项

```
--verbose|-v            #显示详细过程，最多可以 -vvvv
--quiet|-q               #不输出错误信息
--archive|-a             #归档模式，保留文件属性
--recursive|-r           #递归
--backup|-b              #如果目标文件存在，先做备份，默认备份后缀是 ~
--backup-dir=DIR         #指定备份目录
--suffix=SUFFIX          #指定备份后缀
--update|-u              #仅在源mtime比目标已存在文件的mtime新时才拷贝，该选项是接收端判断的，不会影响删除行为
--links|-l              #传输软链接文件，而不是传输其指向的文件
--copy-links|-L          #跟随软链接，传输其指向的目标文件或目录
--perms|-p               #保留权限(不包括特殊权限)
--owner|-o               #保持owner属性
--group|-g               #保持group属性
-D                       #传输设备文件和特殊文件，同 --devices --specials
--times|-t               #保留mtime属性
```

```
--dry-run|-n                #测试模式，不实际传输，常合 -vvvv 配合查看细节
--whole-file|-W             #不使用增量传输，而使用全量传输，在网络带宽高于磁盘带宽时，该选项比
增量传输更高效
--rsh=COMMAND|-e            #指定所要使用的远程shell程序，默认为ssh
--existing                  #只更新远端已存在的文件，远端不存在的文件不传输。使用相对路径时如果
上层目录不存在也不会传输
--ignore-existing           #只传输远程主机不存在的文件
--remove-source-files       #删除已传输成功的源文件
--del                       #在传输中删除，同 --delete-during
--delete                    #文件传输时，删除DEST中在SRC 中不存在的文件
--delete-before             #传输开始前先删除
--delete-during             #在传输过程中删除
--delete-delay              #在传输中确定要删除的文件，传输结束后删除
--delete-after              #传输结束后再删除
--delete-excluded           #删除远程主机上在此项中指定的文件
--force                     #强制删除目录，哪怕不为空
--max-delete=NUM            #最多删除NUM个文件
--max-size=SIZE             #限制rsync传输的最大文件大小，可以使用单位后缀，可以写小数值
--min-size=SIZE             #限制rsync传输的最小文件大小。可用于禁止传输小文件或垃圾文件，可以
写小数值
--size-only                 #默认传输有变化的文件，默认检查文件大小变化和 mtime，此项表示只检
查文件大小变化
--compress|-z               #压缩传输
--exclude=PATTERN           #用正则来指定排除规则来排除不需要传输的文件
--exclude-from=FILE         #从文件中读取不需要被传送的文件
--include=PATTERN           #用正则来指定要传输的文件
--include-from=FILE         #从文件中读取要传输的文件
--port=PORT                 #连接daemon时使用的端口号，默认为873端口
--human-readable|-h         #以人类友好的格式显示相关信息
--progress                  #显示进度条
-P                          #显示进度条，同 --partial --progress
--password-file=FILE        #daemon模式时密码文件,读取密码非交互式操作 这不是远程shell认证的
密码，而是rsync模块认证的密码
--version|-V                    #显示版本信息
--help|-h (*)                   #显示帮助信息
```

1.4.3.3 rsync服务

环境准备

主机清单

IP	角色	系统
10.0.0.208	rsync-server	ubuntu24
10.0.0.206	rsync-client	rocky9

服务端以守护进程运行

```
默认情况下，rsync服务，竟然是没有启动
root@ubuntu24-13:~# systemctl status rsync
○ rsync.service - fast remote file copy program daemon
     Loaded: loaded (/usr/lib/systemd/system/rsync.service; disabled; preset:
enabled)
     Active: inactive (dead)
       Docs: man:rsync(1)
             man:rsyncd.conf(5)
```

```
服务端以守护进程运行
root@ubuntu24-13:~# systemctl enable --now rsync.service
Synchronizing state of rsync.service with SysV service script with
/usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable rsync
Created symlink /etc/systemd/system/multi-user.target.wants/rsync.service →
/usr/lib/systemd/system/rsync.service.
```

默认情况下，服务启动失败

```
root@ubuntu24-13:~# systemctl status rsync
○ rsync.service - fast remote file copy program daemon
     Loaded: loaded (/usr/lib/systemd/system/rsync.service; enabled; preset:
enabled)
     Active: inactive (dead)
  Condition: start condition unmet at Tue 2024-11-26 17:18:16 CST; 13s ago
             └─ ConditionPathExists=/etc/rsyncd.conf was not met
       Docs: man:rsync(1)
             man:rsyncd.conf(5)

11月 26 17:18:16 ubuntu24-13 systemd[1]: rsync.service - fast remote file copy
program daemon was skipped because of an unmet >
lines 1-9/9 (END)
```

```
服务启动失败的原因是 文件不存在
root@ubuntu24-13:~# ls /etc/rsyncd.conf
ls: 无法访问 '/etc/rsyncd.conf': 没有那个文件或目录
```

启动服务

准备对应的文件后，启动服务

```
创建空配置文件
root@ubuntu24-13:~# touch /etc/rsyncd.conf
```

```
启动服务并检查效果
root@ubuntu24-13:~# systemctl start rsync.service
root@ubuntu24-13:~# systemctl status rsync
● rsync.service - fast remote file copy program daemon
     Loaded: loaded (/usr/lib/systemd/system/rsync.service; enabled; preset:
enabled)
     Active: active (running) since Tue 2024-11-26 17:20:11 CST; 2s ago
       Docs: man:rsync(1)
             man:rsyncd.conf(5)
   Main PID: 3790 (rsync)
```

```
      Tasks: 1 (limit: 4558)
     Memory: 840.0K (peak: 1.2M)
        CPU: 33ms
     CGroup: /system.slice/rsync.service
             └─3790 /usr/bin/rsync --daemon --no-detach

11月 26 17:20:11 ubuntu24-13 systemd[1]: Started rsync.service - fast remote file
copy program daemon.
11月 26 17:20:11 ubuntu24-13 rsyncd[3790]: rsyncd version 3.2.7 starting,
listening on port 873
```

```
检查监听端口服务
root@ubuntu24-13:~# ss -tunlp | grep rsync
tcp   LISTEN 0      5            0.0.0.0:873        0.0.0.0:*    users:
(("rsync",pid=3790,fd=4))
tcp   LISTEN 0      5               [::]:873           [::]:*    users:
(("rsync",pid=3790,fd=5))
结果显示：
    该服务监听在 873 端口上。
```

rocky客户端连接测试

```
在客户端进行rsync连接测试，虽然没有信息输出，但是命令是执行成功的
[root@rocky9-12 ~]# rsync rsync://10.0.0.13
[root@rocky9-12 ~]# echo $?
0
原因解读：
    因为服务端的rsync配置文件是空的，没有监听任何信息
```

1.4.3.4 匿名传递

监控链接测试

定制服务端配置

```
在配置文件中，增加相关的配置
root@ubuntu24-13:~# cat /etc/rsyncd.conf
[dir1]
path=/data/dir1/
read only=no
```

```
重启服务
root@ubuntu24-13:~# systemctl restart rsync.service
```

查看服务端目录

```
root@ubuntu24-13:~# tree /data/
/data/
├── dir1
└── dir2

3 directories, 0 files
```

客户端再次测试，可以看到内容

```
连接测试方法1 -- 指明使用rsync协议
[root@rocky9-12 ~]# rsync rsync://10.0.0.13
dir1
```

```
连接测试方法2 --  :: 表示走rsync协议
[root@rocky9-12 ~]# rsync 10.0.0.13::
dir1
```

因为没有配置连接测试用户，所以，默认情况下传输信息回导致失败

```
[root@rocky9-12 ~]# rsync /etc/fstab root@10.0.0.13::dir1
rsync: [receiver] mkstemp "/.fstab.CFHAle" (in dir1) failed: Permission denied
(13)
rsync error: some files/attrs were not transferred (see previous errors) (code
23) at main.c(1330) [sender=3.2.3]
原因解读：
    此处的 root 是指rsync服务的用户，当前服务端并没有配置此信息，默认会被映射成 nobody
```

在服务端配置权限后，客户端继续测试

```
为nobody提供读写执行权限
root@ubuntu24-13:~# setfacl -m u:nobody:rwx /data/dir1/
```

```
客户端再次测试，上传成功,现在使用的是匿名连接
[root@rocky9-12 ~]# rsync /etc/fstab root@10.0.0.13::dir1   # 使用root用户
[root@rocky9-12 ~]# rsync /etc/issue tom@10.0.0.13::dir1    # 使用不存在的tom用户
```

```
服务端查看，属主属组都是 nobody
root@ubuntu24-13:~# ls -l /data/dir1/
总计 8
-rw-r--r-- 1 nobody nogroup 661 11月 26 17:31 fstab
-rw-r--r-- 1 nobody nogroup  23 11月 26 17:32 issue
```

1.4.3.5 指定用户传递

在服务端定制配置，设置应用的用户信息

```
指定映射账号，指定日志文件，指定远程连接用户名和密码，禁用匿名连接
root@ubuntu24-13:~# cat /etc/rsyncd.conf
uid=root
gid=root
max connections=0
log file=/var/log/rsyncd.log
pid file=/var/run/rsyncd.pid
lock file=/var/run/rsyncd.lock

[dir1]
path=/data/dir1/
comment=rsync dir1
read only=no
auth users=rsyncer
secrets file=/etc/rsyncd.pwd
```

指定认证时候专用的密码文件后，重启服务

```
密码文件
root@ubuntu24-13:~# echo 'rsyncer:123456' > /etc/rsyncd.pwd

密码文件权限必须是 600，否则，即使服务启动后，也无法使用

重启服务
root@ubuntu24-13:~# systemctl restart rsync.service
```

客户端再次测试，匿名非匿名都报错

```
使用匿名用户传递信息
[root@rocky9-12 ~]# rsync 10.0.0.13::dir1
Password:
@ERROR: auth failed on module dir1
rsync error: error starting client-server protocol (code 5) at main.c(1821)
[Receiver=3.2.3]
结果显示：
    即使密码正确，也会出现问题。
```

客户端使用指定用户连接测试

```
正常连接的状态
[root@rocky9-12 ~]# rsync rsyncer@10.0.0.13::dir1
Password:
drwxrwxr-x          4,096 2024/11/26 17:32:53 .
-rw-r--r--            661 2024/11/26 17:31:43 fstab
-rw-r--r--             23 2024/11/26 17:32:53 issue
```

```
正常提交文件
[root@rocky9-12 ~]# rsync /etc/hosts rsyncer@10.0.0.13::dir1
Password:

服务端查看效果
root@ubuntu24-13:~# ll /data/dir1
总计 20
drwxrwxr-x+ 2 root   root    4096 11月 26 17:43 ./
drwxr-xr-x  4 root   root    4096 11月 26 16:17 ../
-rw-r--r--  1 nobody nogroup  661 11月 26 17:31 fstab
-rw-r--r--  1 root   root     158 11月 26 17:43 hosts        # 新文件属主属组发生了变
化
-rw-r--r--  1 nobody nogroup   23 11月 26 17:32 issue
```

拓展 -- 如果服务端的密码文件不是 600权限的话，会导致报错

```
客户端测试
[root@rocky9-12 ~]# rsync rsyncer@10.0.0.13::dir1
Password:
@ERROR: auth failed on module dir1
rsync error: error starting client-server protocol (code 5) at main.c(1821)
[Receiver=3.2.3]

服务端可以查看到日志报错的信息
root@ubuntu24-13:~#  cat /var/log/rsyncd.log
...
2024/11/26 17:38:26 [3859] secrets file must not be other-accessible (see strict
modes option)
2024/11/26 17:38:26 [3859] auth failed on module dir1 from UNKNOWN (10.0.0.12)
for rsyncer: ignoring secrets file
```

1.4.3.6 指定密码文件同步

客户端准备环境

```
rocky9 准备相关的文件
mkdir /data/www/dira -p
cp /etc/os-release /data/www/dira/
touch /data/www/{f1,f2}
cp /etc/fstab /data/www/
cp /etc/issur /data/www/
```

```
查看目录效果
[root@rocky9-12 ~]# tree /data/www/
/data/www/
├── dira
│   └── os-release
├── f1
├── f2
├── fstab
└── issue

1 directory, 5 files
```

在客户端，将密码保存到单独的文件，实现非交互式连接

```
[root@rocky9-12 ~]# echo "123456" > /etc/rsyncd.pwd
[root@rocky9-12 ~]# chmod 600 /etc/rsyncd.pwd
```

客户端测试，保证服务端与客户端同步

```
客户端一条命令实现文件的同步效果
[root@rocky9-12 ~]# rsync -avz --delete --password-file=/etc/rsyncd.pwd
/data/www/ rsyncer@10.0.0.13::dir1
sending incremental file list
deleting hosts
./
f1
f2
fstab
```

```
issue
dira/
dira/os-release

sent 685 bytes  received 147 bytes  1,664.00 bytes/sec
total size is 1,191  speedup is 1.43
```

```
命令效果解析
    --delete 选项，此次同步完成后，删除服务端不在此次同步中的文件
    -a 保留属性
    -v 显示过程
    -z 压缩传输
```

查看服务端，确认文件同步效果

```
root@ubuntu24-13:~# tree /data/dir1
/data/dir1
├── dira
│   └── os-release
├── f1
├── f2
├── fstab
└── issue

2 directories, 5 files
结果显示：
    文件同步全部成功
```

增量同步演示

```
客户端增加文件
[root@rocky9-12 ~]# cp /etc/hosts /data/www/
[root@rocky9-12 ~]# rsync -avz --delete --password-file=/etc/rsyncd.pwd
/data/www/ rsyncer@10.0.0.13::dir1
sending incremental file list
./
hosts

sent 333 bytes  received 39 bytes  744.00 bytes/sec
total size is 1,349  speedup is 3.63
```

```
服务端测试效果
root@ubuntu24-13:~# tree /data/dir1
/data/dir1
├── dira
│   └── os-release
├── f1
├── f2
├── fstab
├── hosts
└── issue

2 directories, 6 files
```

1.4.3.7 常用配置

查看配置示例

```
root@ubuntu24-13:~# cat /usr/share/doc/rsync/examples/rsyncd.conf
# sample rsyncd.conf configuration file
# GLOBAL OPTIONS                          #全局配置部份

motd file=/etc/motd                       #motd 文件路径
log file=/var/log/rsyncd                  #日志文件
pid file=/var/run/rsyncd.pid              #PID 文件
syslog facility=daemon                    #rsyslog 日志服务中的日志归类
socket options=                           #socket 选项，具体见 man setsockopt

[ftp]                                     #客户端显示的名称

comment = public archive                  #客户端列出该共享目录时的说明信息
path = /var/www/pub                       #服务端具体目录
use chroot = yes                          #更改映射目录的根目录
max connections=10                        #最大连接数，同时有多少客户端可以连接
lock file = /var/lock/rsyncd              #锁文件
read only = yes                           #默认客户端只读访问
list = yes                                #客户端是否可以列出服务端内容
uid = nobody                              #默认将客户端上传的文件属主映射成 nobody
gid = nogroup                             #默认将客户端上传的文件属组映射成 nobody
exclude =                                 #指定排除项，在此处指定的内容在客户端无法显示
exclude from =                            #从指定文件中读取排除项
include =                                 #指定显示项，配合 exclude,exclude from 使用
include from =                            #同上
auth users =                              #客户端用户名，启用此项表示禁用匿名连接
secrets file = /etc/rsyncd.secrets        #客户端用户名和密码对应文件
strict modes = yes                        #严格检查权限，此选项用于 windows 系统上的
rsync
hosts allow =                             #客户端白名单
hosts deny =                              #客户端黑名单
ignore errors = no                        #不忽略错误
ignore nonreadable = yes                  #
transfer logging = no                     #是否开启传输日志
log format = %t: host %h (%a) %o %f (%l bytes). Total %b bytes.     #默认日志格式
timeout = 600                             #超时时长
refuse options = checksum dry-run         #服务端拒绝客户端执行的命令列表
dont compress = *.gz *.tgz *.zip *.z *.rpm *.deb *.iso *.bz2 *.tbz      #不压缩指
定文件
```

查看配置帮助

```
查看配置帮助
root@ubuntu24-13:~# man rsyncd.conf
```

1.4.4 inotify + rsync 实现数据实时同步

1.4.4.1 定制同步脚本

客户端实现同步的专用脚本

```
准备工作
[root@rocky9-12 ~]# mkdir /data/scrips -p

准备密码文件
[root@rocky9-12 ~]# echo 123456 > /data/scrips/www_rsync.pwd
[root@rocky9-12 ~]# chmod 600 /data/scrips/www_rsync.pwd
```

```
准备脚本文件
[root@rocky9-12 ~]# cat /data/scrips/rsync.sh
#!/bin/bash
# ************************************
# * 功能: Shell脚本模板
# * 作者: 王树森
# * 联系: wangshusen@magedu.com
# * 版本: 2024-11-26
# ************************************

# 定制环境变量
USER="rsyncer"
PASS_FILE="/data/scrips/www_rsync.pwd"
REMOTE_HOST="10.0.0.13"
SRC="/data/www/"
REMOTE_DIR="dir1"
DEST="${USER}@${REMOTE_HOST}::${REMOTE_DIR}"
LOG_FILE="/data/scrips/www_rsync.log"


# 准备工作环境
ubuntu_install_inotify(){
  if [ ! -f /usr/bin/rsync ]; then
    apt install inotify-tools -y
    apt install rsync -y
        fi
}

centos_install_inotify(){
  if [ ! -f /usr/bin/rsync ]; then
    yum install inotify-tools -y
          yum install rsync -y
        fi
}

install_inotify(){
  os_type=$(grep Ubuntu /etc/issue >/dev/null && echo "Ubuntu" || echo "CentOS")
        if [ "${os_type}" == "Ubuntu" ]; then
          ubuntu_install_inotify
        else
          centos_install_inotify
        fi
}

#不间断监控指定目录中的特定事件,当目录中有事件发生变化时，调用 rsync 命令进行同步
rsync_file(){
```

```
  inotifywait -mrq --exclude=".*\.swp" --timefmt '%Y-%m-%d %H:%M:%S' --format
'%T %w %f' -e create,delete,moved_to,close_write,attrib ${SRC} | while read DATE
TIME DIR FILE;do
    FILEPATH=${DIR}${FILE}
          rsync -az --delete --password-file=${PASS_FILE} $SRC $DEST && echo "At
${TIME} on ${DATE}, file ${FILEPATH} was backup via rsync" >> ${LOG_FILE}
  done
}

# 主函数
main(){
  install_inotify
        rsync_file
}

# 执行主函数
main
```

```
查看文件目录
[root@rocky9-12 ~]# tree /data/
/data/
├── scrips
│   ├── rsync.sh
│   ├── www_rsync.log
│   └── www_rsync.pwd
└── www

2 directories, 3 files

执行脚本文件
[root@rocky9-12 ~]# /bin/bash /data/scrips/rsync.sh
# 终端界面阻塞...
```

1.4.4.2 同步测试

在Rocky9客户端新开一个终端，测试效果

```
touch /data/www/f1
cp /etc/fstab /data/www/
dd if=/dev/zero of=/data/www/test.img bs=1M count=10
mkdir -pv /data/www/dira/dirb/dirc
```

ubuntu服务端确认同步效果

```
服务端查看效果
root@ubuntu24-13:~# tree /data/dir1
/data/dir1
├── dira
│   └── dirb
│       └── dirc
├── f1
├── fstab
└── test.img

4 directories, 3 files
```

回到Rocky9客户端，查看同步过程

```
查看客户端日志，
[root@rocky9-12 ~]# cat /data/scrips/www_rsync.log
At 18:43:35 on 2024-11-26, file /data/www/dira/dirb/dir2 was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/f1 was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/f1 was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/f1 was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/fstab was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/fstab was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/test.img was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/test.img was backup via rsync
At 18:45:33 on 2024-11-26, file /data/www/dira was backup via rsync
问题:
    由于某些行为会触发多个事件，导致多次调用脚本，效率不高
```

1.4.5 sersync 实现数据实时同步

1.4.5.1 sersync简介

基础知识

简介

```
    sersync 类似 于inotify，同样用于监控，但它克服了 inotify 的缺点，inotify 最大的不足是会产生重复事件，或者同一个目录下多个文件的操作会产生多个事件，例如，当监控目录中有5个文件时，删除目录时会产生6个监控事件，从而导致重复调用 rsync 命令。另外比如：vim 文件时，inotify 会监控到临时文件的事件，但这些事件相对于 rsync 来说是不应该被监控的。
```

sersync 特点

```
- sersync是使用c++编写，而且对linux系统文件系统产生的临时文件和重复的文件操作进行过滤， 所以在结合rsync同步的时候，节省了运行时耗和网络资源。因此更快。
- sersync 配置很简单，其中提供了静态编译好的二进制文件和 xml 配置文件，直接使用即可
- sersync 使用多线程进行同步，尤其在同步较大文件时，能够保证多个服务器实时保持同步状态
- sersync 有出错处理机制，通过失败队列对出错的文件重新同步，如果仍旧失败，则按设定时长对 同步失败的文件重新同步
- sersync 不仅可以实现实时同步，另外还自带 crontab 功能，只需在 xml 配置文件中开启，即也可以按要求隔一段时间整体同步一次，而无需再额外配置 crontab 功能
- sersync 可以二次开发
```

```
https://code.google.com/archive/p/sersync/          #项目地址
```

1.4.5.2 Rocky9部署sersync

Rocky9获取软件

```
mkdir /data/softs -p ; cd /data/softs
wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/sersync/sersync2.5.4_64bit_binary_stable_final.tar.gz
```

Rocky9解压文件

```
[root@rocky9-12 softs]#  tar xf sersync2.5.4_64bit_binary_stable_final.tar.gz
[root@rocky9-12 softs]#  ls
GNU-Linux-x86  sersync2.5.4_64bit_binary_stable_final.tar.gz
[root@rocky9-12 softs]#  ls GNU-Linux-x86/
confxml.xml  sersync2
```

转移文件到特定目录 -- 该步非必须

```
转移文件
[root@rocky9-12 softs]#  mv GNU-Linux-x86/ /usr/local/sersync
[root@rocky9-12 softs]#  ls /usr/local/sersync/
confxml.xml  sersync2
[root@rocky9-12 softs]#  cd /usr/local/sersync/
```

配置文件说明

```
[root@rocky9-12 sersync]# cat confxml.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<head version="2.5">
    <host hostip="localhost" port="8008"></host>
    <debug start="false"/>                              #是否开启调试模式
    <fileSystem xfs="false"/>
    <filter start="false">                              #不开启文件过滤功能，当为true
时，下列文件不同步
    <exclude expression="(.*)\.svn"></exclude>
    <exclude expression="(.*)\.gz"></exclude>
    <exclude expression="^info/*"></exclude>
    <exclude expression="^static/*"></exclude>
    </filter>
    <inotify>                                               #具体监控的事件
    <delete start="true"/>
    <createFolder start="true"/>
    <createFile start="false"/>
    <closeWrite start="true"/>
    <moveFrom start="true"/>
    <moveTo start="true"/>
    <attrib start="false"/>                                 #此处改为true,则文件属性发生
变化，也会同步
    <modify start="false"/>
    </inotify>

    <sersync>                                               #rsync命令的配置段
    <localpath watch="/opt/tongbu">                         #需要同步的目录
        <remote ip="127.0.0.1" name="tongbu1"/>             #远程主机地址和目录
        <!--<remote ip="192.168.8.39" name="tongbu"/>-->
        <!--<remote ip="192.168.8.40" name="tongbu"/>-->
    </localpath>
    <rsync>
        <commonParams params="-artuz"/>                     #指定rsync选项
        <auth start="false" users="root" passwordfile="/etc/rsync.pas"/>#修改为
true，指定rsync用户名和密码文件,此处的false代表我们要使用的是 ssh协议
        <userDefinedPort start="false" port="874"/>         #指定非标准端口
        <!-- port=874 -->
        <timeout start="false" time="100"/>
        <!-- timeout=100 -->
```

```
        <ssh start="false"/>                                #默认远程rsync daemon运
行,true为使用远程shell
    </rsync>
    <failLog path="/tmp/rsync_fail_log.sh" timeToExecute="60"/>#错误重传及日志文件路
径
    <!--default every 60mins execute once-->
    <crontab start="false" schedule="600"><!--600mins-->#不开启crontab功能
        <crontabfilter start="false">                       #不开启crontab定时传输的筛选
功能
        <exclude expression="*.php"></exclude>
        <exclude expression="info/*"></exclude>
        </crontabfilter>
    </crontab>
    <plugin start="false" name="command"/>
    </sersync>



    #以下不需要修改
    <plugin name="command">
    <param prefix="/bin/sh" suffix="" ignoreError="true"/>  <!--prefix
/opt/tongbu/mmm.sh suffix-->
    <filter start="false">
        <include expression="(.*)\.php"/>
        <include expression="(.*)\.sh"/>
    </filter>
    </plugin>

    <plugin name="socket">
    <localpath watch="/opt/tongbu">
        <deshost ip="192.168.138.20" port="8009"/>
    </localpath>
    </plugin>
    <plugin name="refreshCDN">
    <localpath watch="/data0/htdocs/cms.xoyo.com/site/">
        <cdninfo domainname="ccms.chinacache.com" port="80" username="xxxx"
passwd="xxxx"/>
        <sendurl base="http://pic.xoyo.com/cms"/>
        <regexurl regex="false" match="cms.xoyo.com/site([/a-zA-Z0-
9]*).xoyo.com/images"/>
    </localpath>
    </plugin>
</head>
```

1.4.5.3 配置修改和命令演示

修改配置文件

```
[root@rocky9-12 sersync]# cat confxml.xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<head version="2.5">
    <host hostip="localhost" port="8008"></host>
    <filter start="false">              # 此项不要开启true，否则不会执行文件同步的动作
    ...
    <inotify>
        ...
        <createFile start="true"/>   # 改为 true
```

```
        ...
        <attrib start="true"/>   # 改为 true
        <modify start="true"/>   # 改为 true
    </inotify>

    <sersync>
        <localpath watch="/data/www">   # 本地需要同步的目录
            <remote ip="10.0.0.13" name="dir1"/>  # 需要同步到远程目录
            ...
        </localpath>
        <rsync>
            <commonParams params="-artuz"/>
            # 改为 true，然后使用自定义的rsyncer的用户
            <auth start="true" users="rsyncer" passwordfile="/etc/rsyncd.pwd"/>
            ...
        </rsync>
        ...
```

sersync2 命令用法和参数

```
[root@rocky9-12 sersync]# ./sersync2 -h
set the system param
execute: echo 50000000 > /proc/sys/fs/inotify/max_user_watches
execute: echo 327679 > /proc/sys/fs/inotify/max_queued_events
parse the command param
_____________________________________________________
参数-d:启用守护进程模式
参数-r:在监控前，将监控目录与远程主机用rsync命令推送一遍
c参数-n: 指定开启守护线程的数量，默认为10个
参数-o:指定配置文件，默认使用confxml.xml文件
参数-m:单独启用其他模块，使用 -m refreshCDN 开启刷新CDN模块
参数-m:单独启用其他模块，使用 -m socket 开启socket模块
参数-m:单独启用其他模块，使用 -m http 开启http模块
不加-m参数，则默认执行同步程序
_____________________________________________________________________
```

1.4.5.4 同步实践

准备工作

```
客户端删除所有的无效信息
[root@rocky9-12 ~]# rm -rf /data/www/*

创建依赖密码文件
[root@rocky9-12 ~]# echo 123456 > /etc/rsyncd.pwd
[root@rocky9-12 ~]# chmod 600 /etc/rsyncd.pwd

准备目录结构
[root@rocky9-12 ~]# tree /data/www/
/data/www/
└── dira
    └── dirb
        └── dirc

3 directories, 0 files
```

客户端测试

```
进入到指定目录，然后进行执行
[root@rocky9-12 ~]# cd /usr/local/sersync/
[root@rocky9-12 sersync]# ./sersync2 -dro ./confxml.xml
set the system param
execute: echo 50000000 > /proc/sys/fs/inotify/max_user_watches
execute: echo 327679 > /proc/sys/fs/inotify/max_queued_events
parse the command param
option: -d      run as a daemon
option: -r      rsync all the local files to the remote servers before the
sersync work
option: -o      config xml name:   ./confxml.xml
daemon thread num: 10
parse xml config file
host ip : localhost     host port: 8008
will ignore the inotify createFile event
daemon start, sersync run behind the console
use rsync password-file :
user is rsyncer
passwordfile is         /etc/rsyncd.pwd
config xml parse success
please set /etc/rsyncd.conf max connections=0 Manually
sersync working thread 12  = 1(primary thread) + 1(fail retry thread) +
10(daemon sub threads)
Max threads numbers is: 22 = 12(Thread pool nums) + 10(Sub threads)
please according your cpu , use -n param to adjust the cpu rate
------------------------------------------
rsync the directory recursivly to the remote servers once
working please wait...
execute command: cd /data/www && rsync -artuz -R --delete ./
rsyncer@10.0.0.13::dir1 --password-file=/etc/rsyncd.pwd >/dev/null 2>&1
run the sersync:
watch path is: /data/www
```

检查效果

```
客户端查看进程效果
[root@rocky9-12 sersync]# ps aux | grep confxml
root        2689  0.0  0.0 182464  1536 ?        Ssl  19:33   0:00 ./sersync2 -
dro ./confxml.xml
root        2723  0.0  0.1 221680  2304 pts/1    S+   19:36   0:00 grep --
color=auto confxml
```

```
查看服务端效果
root@ubuntu24-13:~# tree  /data/dir1/
/data/dir1/
└── dira
    └── dirb
        └── dirc

4 directories, 0 files
结果显示：
    文件已经同步成功
```

文件同步测试

```
客户端执行测试命令
[root@rocky9-12 ~]# mkdir -pv /data/www/dirb/dirb/dirc
mkdir: 已创建目录 '/data/www/dirb'
mkdir: 已创建目录 '/data/www/dirb/dirb'
mkdir: 已创建目录 '/data/www/dirb/dirb/dirc'
[root@rocky9-12 ~]# cp /etc/fstab /data/www/
[root@rocky9-12 ~]# dd if=/dev/zero of=/data/www/test.img bs=1M count=10
记录了10+0 的读入
记录了10+0 的写出
10485760字节（10 MB，10 MiB）已复制，0.0170794 s，614 MB/s
```

```
服务端查看
root@ubuntu24-13:~# tree  /data/dir1/
/data/dir1/
├── dira
│   └── dirb
│       └── dirc
├── dirb
│   └── dirb
│       └── dirc
├── fstab
└── test.img

7 directories, 2 files
```