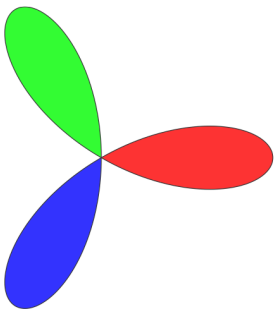
Unix/Linux 工具箱

必知必会的常用工具

作者：凡云

时间：March 3, 2023

版本：第一版



工欲善其事必先利其器

目录

前言

作为一名 Linux 软件工程师，深知工具对我们至关重要.

这本书会列出一些常见的工具的用法和示例，方便大家参考。本书的特色是尽量通过一些实例，来介绍工具的使用方法。本书的目的让大家了解和熟悉相关工具，适合于 Linux 软件工程师入门者参考，为 Linux Shell 编程打下基础。

先看看我们的工具箱应该放哪些工具:

★ 查找、搜寻工具 find
★ 字符串处理工具 awk/sed/grep/hexdump/xxd
★ 压缩工具 tar/zip/7z
★ 文件系统常用工具 fdisk/mkfs/mount
★ 安全相关工具 ssh/scp/rsync/md5sum/sha
★ 网络相关工具 wget/curl
★ 多媒体相关工具 ffmpeg

仅以此书献给所有 Linux 初学者。

第 1 章 find

准备写这本书的时候，find 是我想起来的第一个工具。find 是 UNIX/Linux 中最常用的命令之一，它可以根据文件名、文件大小、文件权限、文件日期和文件类型等进行搜索并可以执行指定操作。

1.1 依据文件类型查找

选项名字：-type, 支持类型如下：

- b 块设备
- c 字符设备
- d 目录
- p 命名管道
- f 普通文件
- l 符号链接
- s 套接字

1. 查找当前目录下所有符号链接

```
find . -type l
```

2. 查找当前目录下所有普通文件

```
find . -type f
```

1.2 依据文件名查找

选项名字: -name

1. 查找当前目录名字为 cat.c 的文件

```
find . -name cat.c
```

2. 查找当前目录所有文件名后缀为.c 或者.h 的文件

```
find . -name "*.[ch]"
```

3. 查找当前目录下名字为 picture 的目录

```
find . -name picture -type d
```

1.3 依据大小查找

选项名字: size，支持的单位选项如下：

- *b* 以 block （512 bytes）为单位
- *c* 以 byte 为单位
- *w* 以 2 bytes 为单位
- *k* 以 1024 bytes 为单位
- *M* 以 MB（1024 * 1024 bytes）为单位
- *G* 以 GB (1024 * 1024 * 1024 bytes) 为单位

1. 查找当前目录 size 为 1024 bytes 的普通文件

```
find . -type f -size 1024c
```

2. 查找当前目录 size 大于 1GB 的普通文件

```
find . -type f -size +1G
```

3. 查找当前目录 size 小于 512M 的普通文件

```
find . -type f -size -512M
```

1.4 依据时间查找

1. 查找当前目录 10 分钟前访问的文件

```
find . -type f -amin +10
```

2. 查找当前目录最近 10 分钟以内访问的文件

```
find . -type f -amin -10
```

3. 查找当前目录 100 分钟前修改的文件

```
find . -type f -cmin +100
```

4. 查找当前目录最近 100 分钟以内修改问的文件

```
find . -type f -cmin -100
```

5. 查找当前目录下修改时间比 *cat.c* 新的文件

```
find . -type f -cnewer ./cat.c
```

1.5 依据文件权限查找

1. 查找当前目录下权限标志位是 664 的文件 (文件 Mode = 0664)

```
find . -perm 0664
```

2. 查找当前目录下权限标志位满足拥有者、同组用户以及拥有者都可以读的文件 (写和可执行权限不在检查范围)

```
find . -perm -0444
find . -perm -u=r,g=r,o=r
```

3. 查找当前目录下权限标志位满足拥有者或者同组用户可以写的文件

```
find . -perm /0220
find . -perm /u=w,g=w
```

1.6 查找后执行相关操作

1. 在 /tmp 目录查找后缀为.log 的文件并删除

```
find /tmp -name "*.log" | xargs rm -f
find /tmp -name "*.log" -delete
find /tmp -name "*.log" -exec rm -f '{}' \;
```

注: 当文件命中有空格是, find 搭配 xargs 删除文件会出错.

2. 在当前目录查找后缀为.mp3 的文件，并转换为.wav 文件

```
find . -name *.mp3 -exec ffmpeg -i '{}' '{}'.wav \;
```

1.7 多条件联合查找-全部满足

查找当前目录下所有用户都可以读写当不能执行的文件:

```
find . -perm -444 -perm /222 \! -perm /111
```

1.8 多条件联合查找-满足部分

查找当前目录文件大小大于 8MB 或者小于 10KB 的文件:

```
find . \( -size +8M -o -size -10k \) -type f
```

1.9 指定搜索目录层级

1. 查找 /var/log 目录下名字后缀为.log 的文件，忽略下面的子目录

```
find /var/log/ -maxdepth 1 -type f -name "*.log"
```

2. 查找 /var/log 目录下名字后缀为.log 的文件，子目录最大深度为 2

```
find /var/log/ -maxdepth 2 -type f -name "*.log"
```

3. 查找 /var/log 目录下名字后缀为.log 的文件，子目录最小深度为 2

```
find /var/log/ -mindepth 2 -type f -name "*.log"
```

1.10 总结

初学者可以忽略本节。

基本用法: find [-H] [-L] [-P] [-D debugopts] [-Olevel] [starting-point...] [expression]

★ *-H -L -P* 这三个选项指定搜索时是否忽略符号链接

-P 忽略符号链接

-L 跟随符号链接

-H 忽略符号链接，除非它出现在命令行参数 starting-point 中

★ *-D debugopts* debug 选项，不常用；MacOS 版本没有这个选项

★ *-Olevel* 优化选项

★ *expression* 有 5 种表达式，分别是 Tests, Actions, Global Options, Positional Options, Operators

- *Tests* 设定搜索条件，比如 -size 或 -type 等
- *Actions* 设定满足条件后执行的动作，比如 -print 或 -exec 等
- *Global Options* 设定全局搜索条件，比如 -maxdepth 或 -mindepth. 注意 *Global Options* 和 *Tests* 的区别，后者是指赛选出来的内容必须满足的条件，而前者更多是指在搜索之前限定的一些条件；比如仅搜索当前目录下的文件，忽略其他子目录等，需要设定 *Global Options*，而非 *Tests*; 注意 *Global Options* 表达式要跟在 *start-point...* 的后面，不能在其他位置。
- *Positional Options* 放在 *start-point...* 的后面，影响后面所有 *Tests* 或 *Actions*, 比如 -daystart 参数。注：关于 daystart 参数，在线手册描述的不够清楚，。初学者可以忽略这些表达式。
- *Operators* 与或非运算符, 运算符表达式尽量要用括号包起来。

  and -a 或者 -and

  or -o 或者 -or

  not ！或者 -not

第 2 章 压缩解压文件常用命令

工作和学习中我们会经常会收到不同后缀的压缩包，.bz2, .7z, .zip 等。了解本章的三个工具，可以让我们轻松应对各种压缩格式。

2.1 tar

tar 是一个常用归档工具，它可以将一个或多个文件打包在一起，从而节省空间，方便传输。tar 命令的语法如下：

基本用法：tar [子命令] -f [归档文件名] [选项] [文件列表]

*归档文件名*指的是归档后的文件名；*文件列表*是指待归档的文件或目录。

tar 本身不支持压缩或解压缩，但可以调用其他压缩工具来执行压缩或解压动作；可以通过参数设定压缩工具的名称。

tar 支持的操作或者子命令

★ *-A* 将一个 tar 包追加到另外一个 tar 包的后面
★ *-c* 生成新的 tar 包，将多个文件归档在一起
★ *-d* 比较 tar 包和文件系统的差异
★ *-t –list*，罗列 tar 包中的内容
★ *-r* 追加新的文件到 tar 包中
★ *-u* 如果参数中的文件比 tar 包中的新，则更新 tar 包中的文件
★ *-x –extract* 从 tar 中提取文件，可部分提取

tar 支持的常用选项

★ *-a* 根据文件名后缀自动推导所需压缩工具的名字
★ *-v* 显示处理每个文件的信息
★ *-f* tar 包文件名
★ *-j* 指定压缩工具为 bzip2
★ *-J* 指定压缩工具为 xz
★ *–lzip* 指定压缩工具为 lzip
★ *–lzma* 指定压缩工具为 lzma
★ *–lzop* 指定压缩工具为 lzop
★ *-z* 指定压缩工具为 gzip
★ *-Z* 指定压缩工具为 compress
★ *-C –directory* 设定工作目录
★ *-T, –files-from* 从指定文件中获取要归档或提取的文件列表，而非命令行参数

tar 常用示例

1. 将 data 目录归档在一起, 生成 tar 包 data.tar, 但不压缩

```
# tar cvf data.tar ./data/
```

2. 解压到指定目录，比如将 data.tar 解压到 /Downloads 目录

```
# tar xvf data.tar -C ~/Downloads/
```

3. 将 data 目录归档并采用 bzip2 压缩, 生成压缩包 data.tar.bz2

```
# tar jcvf data.tar.bz2 ./data/
```

上述命令略等与以下两个命令:

```
# tar cvf data.tar ./data/
# bzip2 -z data.tar
```

对应解压命令为:

```
# tar jxvf data.tar.bz2
```

4. 将 data 目录归档并采用 zip 压缩, 生成压缩包 data.tar.gz

```
# tar zcvf data.tar.gz ./data
```

上述命令略等与以下两个命令:

```
# tar cvf data.tar ./data/
# gzip data.tar
```

对应解压命令为:

```
# tar zxvf data.tar.gz
```

5. 将 data 目录归档并采用 xz 压缩, 生成压缩包 data.tar.xz

```
# tar Jcvf data.tar.xz ./data
```

上述命令略等与以下两个命令:

```
# tar cvf data.tar ./data/
# xz data.tar
```

对应解压命令为:

```
# tar Jxvf data.tar.xz
```

6. 将 data 目录归档并采用 lzip 压缩, 生成压缩包 data.tar.lz

```
# tar cvf data.tar.lz --lzip ./data
```

上述命令略等与以下两个命令:

```
# tar cvf data.tar ./data/
# lzip data.tar
```

对应解压命令为:

```
# tar xvf data.tar.lz --lzip
```

7. 将 data 目录归档并采用 lzma 压缩, 生成压缩包 data.tar.lzma

```
# tar cvf data.tar.lzma --lzma ./data
```

上述命令略等与以下两个命令:

```
# tar cvf data.tar ./data/
# lzma data.tar
```

对应解压命令为:

```
# tar xvf data.tar.lzma --lzma
```

8. 将 data 目录归档并采用 lzop 压缩, 生成压缩包 data.tar.lzo

```
# tar cvf data.tar.lzo --lzop ./data
```

上述命令略等与以下两个命令:

```
# tar cvf data.tar ./data/
# lzop data.tar
```

对应解压命令为:

```
# tar xvf data.tar.lzo --lzop
```

9. 不指定压缩工具，让 tar 根据文件名后缀自动推断使用何种压缩工具 (-a 选项)

```
# tar acvf data.tar.xz ./data  # 呼叫 xz
# tar acvf data.tar.bz2 ./data # 呼叫 bzip2
# tar acvf data.tar.lz ./data  # 呼叫 lzip
# tar acvf data.tar.lzma ./data # 呼叫 lzma
# tar acvf data.tar.lzo ./data # 呼叫 lzop
```

10. 不指定压缩工具，让 tar 根据文件名后缀自动解压 (-a 选项)

```
# tar axvf data.tar.xz  # 呼叫 xz
# tar axvf data.tar.bz2 # 呼叫 bzip2
# tar axvf data.tar.lz  # 呼叫 lzip
# tar axvf data.tar.lzma # 呼叫 lzma
# tar axvf data.tar.lzo # 呼叫 lzop
```

11. 将 test2.tar 追加到 test.tar

```
# tar -Af test.tar test2.tar
```

12. 追加新的文件 mem.img 到 tar 包 data.tar, 不支持向压缩包中的添加新文件

```
# tar rvf data.tar mem.img
```

13. 查看压缩包 data.tar.bz2 中的内容

```
# tar jtvf data.tar.bz2
```

14. 从压缩包 data.tar.bz2 中解压指定文件 *./data/all.txt*, 并存储到指定目录 */Downloads*

```
# tar jxvf data.tar.bz2 -C ~/Downloads/ ./data/all.txt
```

注意：如果待提取的文件位于某个目录下面，目录名要和 *tar -t* 查看到的信息一致；比如 *./data/all.txt* 不能写为 *data/all.txt*

15. 从指定文件 list.txt 中获取文件列表，执行归档和压缩

```
#tar jcvf data_from_list.tar.bz2 -T list.txt
./data/all.txt
./data/lower.txt
./data/time.txt
./img/ramdon.img
./img/sda_1st.img
./img/sde3.img
```

其中，list.txt 的内容如下:

```
#cat list.txt
./data/all.txt
./data/lower.txt
./data/time.txt
./img/ramdon.img
./img/sda_1st.img
./img/sde3.img
```

2.2 7z

linux 上的 7z 是一个可以同时执行归档和压缩解压缩的工具，支持 7z、LZMA2, XZ, ZIP, Zip64, CAB, RAR, ARJ, GZIP, BZIP2 等压缩格式。和 tar 不同，tar 只负责归档，压缩解压功能由其他工具实现。但 7z 不能保存原有文件的权限标志位信息和组用户信息。

基本用法: 7z 子命令选项压缩文件名待压缩文件列表

子命令

- ★ *a* 增加文件到压缩包; 压缩包不存在，则直接创建；否则追加到压缩包
- ★ *d* 从压缩包删除文件
- ★ *e* 从压缩包提前文件
- ★ *h* 计算文件哈希
- ★ *i* 罗列支持的格式
- ★ *l* 列出压缩包中的文件列表
- ★ *rn* 重命名压缩包中的文件
- ★ *t* 验证压缩包完整性
- ★ *u* 更新压缩包中的文件
- ★ *x* 从压缩包中提取文件，保持完整路径

常用选项

- ★ *-m* 设定压缩方法, 改参数和 *-t* 相关，每个类型对应的 m 参数不相同
- ★ *-p* 设定密码
- ★ *-sfx* 创建 sfx 归档文件 (Widows 使用，可以自解压)
- ★ *-t* 设定归档文件类型, 支持：*, #, 7z, xz, split, zip, gzip, bzip2, tar 等。默认从文件名后缀推导

常用示例

1. 将当前目录下.txt 文件，归档到 text.zip, 不压缩;

```
# 7z a text.zip ./*.txt -mx0
```

2. 将当前目录下.txt 文件，归档到 text.zip, 采用 lzma 压缩方法，压缩级别 9 (Untrla)

```
# 7z a -tzip test.zip ./*.txt  -mx9 -mm=LZMA
```

3. 将当前目录下.txt 文件，归档到 text.7z, 采用 lzma2 压缩方法，压缩级别 9 (Untrla)

```
# 7z a -t7z test.7z ./*.txt  -mx9 -m0=LZMA2
```

4. 查看 text.zip 中的内容

```
# 7z l text.zip
```

5. 从压缩包 text.zip 中提取所有文件

```
# 7z x text.7z
```

6. 从压缩包 text.zip 中提取文件 data/all.txt
   提取文件的同时，保持目录结构:

```
# 7z x text.7z data/all.txt
```

   同上条命令，但仅提取文件，不保持目录结构:

```
# 7z e text.7z data/all.txt
```

7. 从压缩包 text.zip 中删除文件 pic.rgb

```
# 7z d text.zip pic.bgr
```

8. 检查压缩包的完整性

```
# 7z t text.7z
```

9. 加密待归档文件，将 m0.zip 以加密的方式添加到 text.zip，密码为 123456

```
# 7z a -p123456 text.7z m0.zip
```

   从 text.zip 提取加密文档 m0.zip

```
# 7z x -p123456 text.7z m0.zip
```

10. 制作自解压包

```
# 7z a -sfx archive.exe dir1
```

2.3 zip

zip 是一个打包和压缩工具，支持 Windows，Unix，MacOS 等多个操作系统。
基本用法: zip 选项压缩包名称输入文件列表

常用选项

- ★ *-d* 从压缩包删除文件
- ★ *-e* 打开加密功能
- ★ *-i, –include*
- ★ *-m* 归档后删除源文件
- ★ *-n* 指定不压缩文件
- ★ *-r, –recurse-path*
- ★ *-R, –recurse-patterns*
- ★ *-T* 验证压缩包完整性
- ★ *-u, –update*
- ★ *-x, –exlucde*
- ★ *-Z, –compression-method* 指定压缩方法, 支持 store, deflate 和 bzip2
- ★ *-q* 安静模式，不输出 log
- ★ *-#* 指定压缩级别, 0-9; 0 是不压缩，9 是压缩比最高

常用示例

1. 打包当前目录所有.c 和.h 文件, 压缩生成 data.zip

```
# zip data.zip *.c *.h
```

2. 递归打包和压缩 data 目录，生成压缩包 data.zip

```
# zip -r data.zip ./data
```

如果期望压缩之后删除源文件，可以加 -m 选项

```
# zip -rm data.zip ./data
```

3. 从 data.zip 中删除文件 data/all.txt

```
# zip -d data.zip data/all.txt
```

4. 递归打包和压缩 data 目录，生成压缩包 data.zip, 对.rgb 和.bgr 文件不执行压缩

```
# zip -r -n .bgr:.rgb data.zip ./data/
```

5. 递归打包和压缩 data 目录，生成压缩包 data.zip, 但将.jpg 和.jpeg 排除在外

```
# zip -r  data.zip ./data/ -x \*.jpg \*.jpeg
```

6. 递归打包和压缩 data 目录，生成压缩包 data.zip, 指定压缩方法为 bzip, 压缩级别选择 9

```
# zip -9 Z bzip2 -r data.zip ./data
```

7. 递归打包和压缩 data 目录，生成压缩包 data.zip, 打包文件从 list.txt 选择, 而非全部

```
# zip -r data.zip ./data  -i@list.txt
```

8. 递归打包和压缩 data 目录，生成压缩包 data.zip, 压缩之后加密压缩包
加密过程会提示输入密码并确认:

```
# zip -e -r data.zip ./data
Enter password:
Verify password:
```

解压时会提示输入密码:

```
# unzip data.zip
Archive:  data.zip
[data.zip] data/lower.txt password:
```

9. 验证 data.zip 的完整性

```
# zip -T data.zip
```

第 3 章 文件系统常用命令

3.1 fdisk

fdisk 是 Linux 系统的磁盘管理工具，它可以用来查看磁盘分区，创建、删除、重新定位或格式化磁盘分区。fdisk 的基本功能需要进入交互式模式才可以使用。

下面以操作一个 128GB 的 U 盘为例，进行举例说明。在笔者的 Ubuntu 20.04 系统上，插入 U 盘后，系统多了一个 /dev/sde 节点。当然在其他系统上，可能会是其他节点, 比如 /dev/sdb 或 /dev/sdc 等。

1. 插入 U 盘之前，执行 ls /dev/sd*

```
#ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sda4  /dev/sda5  /dev/sda6
```

2. 插入 U 盘之后，执行 ls /dev/sd*

```
#ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda3  /dev/sda4  /dev/sda5  /dev/sda6   /dev/sde
```

3. 执行 sudo fdisk -l /dev/sde 查看 U 盘基本信息

```
# sudo fdisk -l /dev/sde
Disk /dev/sde: 114.6 GiB, 123060879360 bytes, 240353280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

4. 执行 fdisk 命令进入交互模式; fdisk 在交互模式下支持一系列命令，每个命令以单个字母命名；输入命令后，敲入回车即执行。

```
#sudo fdisk /dev/sde
Welcome to fdisk (util-linux 2.31.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x1577f626.
```

5. m 查看 help 信息 (输入字母 m，然后敲回车，后续命令同理)

```

Command (m for help): m

Help:

   DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag

   Generic
```

```
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition

   Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)

   Script
   I   load disk layout from sfdisk script file
   O   dump disk layout to sfdisk script file

   Save & Exit
   w   write table to disk and exit
   q   quit without saving changes

   Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table
```

6. p 查看分区信息

```
Command (m for help): p
Disk /dev/sde: 114.6 GiB, 123060879360 bytes, 240353280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1577f626
```

7. n 新增第一个分区（注：新增分区命令会让用户选择分区类型、起始扇区和结束扇区或分区大小）
   - 新增一个 primary 分区，分区号为 1
   - 分区起始 sector 为 2048 (前面预留了一些 sector，用来做分区管理和引导扇区)
   - 分区大小设定为 32G

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-240353279, default 2048): 2048
```

```
Last sector, +sectors or +size{K,M,G,T,P} (2048-240353279, default 240353279): +32G

Created a new partition 1 of type 'Linux' and of size 32 GiB.
```

8. p 查看分区信息

```
Command (m for help): p
Disk /dev/sde: 114.6 GiB, 123060879360 bytes, 240353280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1577f626

Device     Boot Start      End  Sectors Size Id Type
/dev/sde1        2048 67110911 67108864 32G 83 Linux
```

9. n 新增一个扩展分区
   - 新增一个 extended 分区，分区号为 2
   - 分区起始 sector 为 67110912
   - 分区大小设定为 64G

```
Command (m for help): n
Partition type
   p   primary (1 primary, 0 extended, 3 free)
   e   extended (container for logical partitions)
Select (default p): e
Partition number (2-4, default 2): 2
First sector (67110912-240353279, default 67110912):
Last sector, +sectors or +size{K,M,G,T,P} (67110912-240353279, default 240353279): +64G

Created a new partition 2 of type 'Extended' and of size 64 GiB.
```

10. p 查看分区信息

```
Command (m for help): p
Disk /dev/sde: 114.6 GiB, 123060879360 bytes, 240353280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1577f626

Device     Boot    Start       End   Sectors Size Id Type
/dev/sde1           2048  67110911  67108864  32G 83 Linux
/dev/sde2       67110912 201328639 134217728 64G  5 Extended
```

11. n 基于扩展分区，新增一个逻辑分区
    - 新增一个逻辑分区，分区号为 5
    - 分区起始 sector 为 67112960
    - 分区大小设定为 64G

```
Command (m for help): n
Partition type
   p   primary (1 primary, 1 extended, 2 free)
   l   logical (numbered from 5)
Select (default p): l

Adding logical partition 5
First sector (67112960-201328639, default 67112960):
Last sector, +sectors or +size{K,M,G,T,P} (67112960-201328639, default 201328639):

Created a new partition 5 of type 'Linux' and of size 64 GiB.
```

12. p 查看分区信息

```
Command (m for help): p
Disk /dev/sde: 114.6 GiB, 123060879360 bytes, 240353280 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1577f626

Device     Boot    Start       End   Sectors Size Id Type
/dev/sde1           2048  67110911  67108864  32G 83 Linux
/dev/sde2       67110912 201328639 134217728 64G  5 Extended
/dev/sde5       67112960 201328639 134215680 64G 83 Linux
```

13. w 保存分区并退出 fdisk 交互模式

```
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

14. 执行 ls /dev/sd* 查看系统 block 设备, 我们会发现系统多了 /dev/sde1, /dev/sde2 以及 /dev/sde5 三个分区设备。

```
#ls /dev/sde*
/dev/sde  /dev/sde1  /dev/sde2 /dev/sde5
```

3.2 mkfs

mkfs 工具用于格式化 block 设备或者其上面的分区，是一些列工具的总称。支持的文件系统包括 ext2, ext3, ntfs, ubifs 等。

1. 格式化/dev/sde1 分区为 ext4 文件系统

```
#sudo mkfs -t ext4 /dev/sde1
mke2fs 1.44.1 (24-Mar-2018)
Creating filesystem with 8388608 4k blocks and 2097152 inodes
Filesystem UUID: c6d3d4cd-f876-4702-9cd5-6133da0b9930
Superblock backups stored on blocks:
```

```
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000, 7962624

Allocating group tables: done
Writing inode tables: done
Creating journal (65536 blocks): done
Writing superblocks and filesystem accounting information:
```

2. 格式化/dev/sde5 分区为 exfat 文件系统到

```
#sudo mkfs -t exfat /dev/sde5
mkexfatfs 1.2.8
Creating... done.
Flushing... done.
File system created successfully.
```

mkfs.T 等价于 mkfs -t T

3.3 mount

mount 命令是 Linux 系统中一个非常重要的命令，它用于挂载一个文件系统，使得用户可以在系统中访问该文件系统。

SYNOPSIS: mount [-t 文件系统类型] [-o 选项] 设备文件或文件系统路径挂载点

★ *-t* 参数用于指定要挂载的文件系统类型，比如 ext2、ext3、vfat 等

★ *-o* 参数用于指定挂载的选项，比如 rw、ro 等

★ 设备文件或文件系统路径指的是要挂载的设备文件或文件系统

★ 挂载点指的是挂载的目录，挂载之后可以在此目录下访问挂载的文件系统。

常见 mount 示例

1. 挂载 /dev/sda1 到 /media/datum, 文件系统类型为 ext4

```
mount -t ext4 /dev/sda1 /medai/datum
```

2. 挂载 /dev/sda1 到 /media/datum, 不指定文件系统类型

```
mount /dev/sda1 /medai/datum
```

3. 挂载 NFS 10.0.1.19:/home/cloudnie/nfs 到 /mnt

```
mount -t nfs 10.0.1.19:/home/cloudnie/nfs /mnt
```

4. 挂载 NFS 10.0.1.1:/home/cloudnie/nfs 到 /mnt, 不开启 NLM lock(NFS2 和 NFS3 支持的功能). 如果 NFS server 不是 NFS2 或者 NFS3 版本，必须加 nolock 选项。

```
mount -t nfs -o nolock 10.0.1.19:/home/cloudnie/nfs /mnt
```

5. 挂载 cifs 到 /media/server, 远程站点为 10.0.1.19/backup, 用户名:cloudnie, 密码:MYPASSWD

```
mount -t cifs //10.0.1.19/backup -o username=cloudnie,pass="MYPASSWD" /media/dataset
```

6. 挂载 proc 文件系统到 /var/proc

```
mount -t proc proc /var/proc
```

7. 挂载 tmpfs 文件系统到 /var/tmpfs

```
mount -t tmpfs tmpfs /var/tmpfs
```

8. 挂载 sysfs

```
mount -t sysfs sysfs /var/sysfs/
```

9. 挂载 loop devcie /dev/loop13 到 /var/data

```
mount /dev/loop13 /var/data
```

10. 挂载镜像文件 data.img, mount 会自动绑定 loop device。注意 data.img 必须是一个包含文件系统的镜像，否则需要先建立 loop 设备，然后执行格式化操作。

```
mount data.img /mnt
```

11. 重新挂载 /media/data, 可读写

```
mount -o remount,rw /media/data
```

12. 重新挂载 /media/data, 只读

```
mount -o remount,ro /media/data
```

13. 将目录 /mnt/nfs bind 到 /usr/bin

```
        mount --bind /mnt/nfs /usr/bin
```

注意：通常情况下，执行 mount 命令需要 root 权限；所以上述例子以非 root 用户执行时，需要加 sudo 来提升权限。

3.4 dd

dd 是一个 block 设备的复制工具，可以用来转换或复制文件，能够精准实现基于 block 设备的以字节、块等各种单位的拷贝和转换。

基本用法: dd OPTIONS

常用选项:

★ *if* 指定输入文件
★ *of* 指定输出文件
★ *ibs* 每次从输入文件读取的字节数
★ *obs* 每次向输出文件写入的字节数
★ *bs* 每次读写字节数，会重置 ibs 和 obs
★ *seek* 设定输出文件的偏移，单位为 block，block 大小由 obs 指定
★ *skip* 设定输入文件的偏移，单位为 block，block 大小有 ibs 指定
★ *count* 指定读写的 block 数目，block 大小 ibs 指定
★ *iflag* 指定读文件的参数
★ *oflag* 指定写文件的参数

常用示例

1. 生成一个内容全 0 文件，大小为 128M，文件名 0.img

```
# dd if=/dev/zero of=0.img bs=1M count=128
128+0 records in
128+0 records out
134217728 bytes (134 MB, 128 MiB) copied, 0.0709941 s, 1.9 GB/s
```

2. 生成一个内容随机的文件，大小为 128M，文件名 random.img

```
# dd if=/dev/urandom of=ramdon.img bs=1M count=128
128+0 records in
128+0 records out
134217728 bytes (134 MB, 128 MiB) copied, 1.4226 s, 94.3 MB/s
```

3. 生成硬盘镜像文件

```
#sudo dd if=/dev/sda of=/backup/disk.img
```

4. 从文件 random.img 4MB 开始，copy 2MB 内容生成新的文件 part2.img

```
dd bs=1M skip=4 if=random.img of=part2.img count=2
2+0 records in
2+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.0338895 s, 61.9 MB/s
```

5. 从文件 sde3.img 2MB 开始，copy 3MB 内容，写向 ramdom.img 的 4MB 处

```
# dd bs=1M skip=2 if=sde3.img seek=4 of=ramdon.img count=3
3+0 records in
3+0 records out
3145728 bytes (3.1 MB, 3.0 MiB) copied, 0.0327214 s, 96.1 MB/s
```

6. 备份分区 /dev/sde3, 文件名为 sde3.img

```
#sudo dd if=/dev/sde3 of=sde3.img
262144+0 records in
262144+0 records out
134217728 bytes (134 MB, 128 MiB) copied, 4.58755 s, 29.3 MB/s
```

7. 将备份文件 sde3.img 恢复到设备 /dev/sde3

```
#sudo dd if=./sde3.img of=/dev/sde3
262144+0 records in
262144+0 records out
134217728 bytes (134 MB, 128 MiB) copied, 28.1431 s, 4.8 MB/s
```

8. 备份硬盘 sda 的第一个扇区

```
#sudo dd if=/dev/sda of=sda_1st.img bs=512 count=1
1+0 records in
1+0 records out
512 bytes copied, 0.0444258 s, 11.5 kB/s
```

9. 从 ISO 文件创建一个 USB 闪存驱动器启动盘

```
#sudo dd if=ubuntu-22.04.2-desktop-amd64.iso of=/dev/sde bs=1M
1767+1 records in
1767+1 records out
1853431808 bytes (1.9 GB, 1.7 GiB) copied, 149.775 s, 12.4 MB/s
```

10. 复制一个硬盘到另一个硬盘

```
#sudo dd if=/dev/sdb of=/dev/sdc
```

11. 将磁盘分区 sde3 备份到 sde4,

```
#sudo dd if=/dev/sde3 of=/dev/sde4 bs=4K
32768+0 records in
32768+0 records out
```

12. 将多个文件合并到一个文件中

```
# dd conv=notrunc if=upper.txt oflag=append of=all.txt
0+1 records in
0+1 records out
29 bytes copied, 0.00019355 s, 150 kB/s

# dd conv=notrunc if=lower.txt oflag=append of=all.txt
0+1 records in
0+1 records out
29 bytes copied, 0.000156844 s, 185 kB/s
```

13. 将内存中的内容写入文件；处于安全考虑比较新的 kennel 通常会限制 user mode 任意访问 /dev/mem；在 ubuntu 20.04 上，这条命令执行失败。

```
#sudo dd if=/dev/mem of=mem.img bs=4M count=4
dd: error reading '/dev/mem': Operation not permitted
0+0 records in
0+0 records out
0 bytes copied, 0.0580443 s, 0.0 kB/s
```

14. direct IO, 直接写入存储设备，不经过 cache

```
#sudo dd if=sde3.img of=/sde oflag=direct
262144+0 records in
262144+0 records out
134217728 bytes (134 MB, 128 MiB) copied, 23.1987 s, 5.8 MB/s
```

15. 将文件中的小写字母全部转为大写字母

```
# dd conv=ucase if=time.txt of=tmp.txt
0+1 records in
0+1 records out
29 bytes copied, 0.000156911 s, 185 kB/s
```

16. 将文件中的大写字母全部转为小写字母

```
# dd conv=lcase if=time.txt of=tmp.txt
0+1 records in
0+1 records out
29 bytes copied, 0.000156911 s, 185 kB/s
```

17. 交换 Planar RGB 文件 pic.rgb 三个通道的顺序，由 RGB 变为 BGR, 生成新文件 pic.bgr. 假定每个通道大小为 1MB.

```
# dd bs=1M skip=2 seek=0 conv=notrunc oflag=append count=1 if=pic.rgb of=pic.bgr
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00166178 s, 631 MB/s
# dd bs=1M skip=1 seek=1 conv=notrunc oflag=append count=1 if=pic.rgb of=pic.bgr
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00136352 s, 769 MB/s
# dd bs=1M skip=0 seek=2 conv=notrunc oflag=append count=1 if=pic.rgb of=pic.bgr
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.00105084 s, 998 MB/s
```

3.5 常见文件操作示例

1. 创建目录

```
mkdir /media/data /media/datum #在目录 /media 下创建子目录 data 和 datum
mkdir /media/data/audio/wav #在目录 /media 下创建子目录 data/audio/wav, 如果中间目录不存在则直接创
    建
```

2. 删除目录, 下面两条命令的作用是一样的

```
rm -rf /media/data #删除目录 /media/data, 包括其下面的子目录
rmdir /media/data #删除目录 /media/data; 如果目录不为空, 则给出 warning
rm -d /media/data # 和上一条命令等价
```

3. 创建空文件

```
touch fs.c #在当前目录创建新文件 fs.c; 如果 fs.c 存在则修改其访问时间和修改时间为当前时间
touch -d "2014-12-26 14:20:55" fs.c #同上, 以-d 后面的参数代替当前时间
touch -t 201412261420.55 fs.c #同上, 以 -t 后面的参数代替当前时间
```

4. 删除文件

```
rm -f fs.c #删除文件 fs.c, 不给出提示或者警告信息
```

5. 移动或重命名文件

```
mv fs.c filesystem.c #将 fs.c 重命名为 filesystem.c
mv fs.c ../ #将 fs.c 移动到上级目录
```

6. 更改文件权限标志位

```
chmod a+x test.sh # 修改 test.sh 的权限标志位，增加所有用户可执行权限
chmod ugo+x test.sh # 同上
chmod a-w test.sh # 修改 test.sh 的权限标志位，去掉所有用户的写权限
chmod ugo-w test.sh # 同上
chmod 0777 test.sh # 设定 tesh.sh 的文件权限位为 0777， 即所有用户可读写，可执行
```

7. 更改文件拥有者属性

```
chown cloudnie:cloudnie test.sh # 修改 test.sh 的拥有者和组分别为 cloudnie 和 cloudnie
chown -R cloudnie:cloudnie /media/data # 修改目录 /meida/data 的拥有者和组分别为 cloudnie 和
    cloudnie，
           如果 /media/data 下有子目录，则递归修改
```

8. 拷贝文件和目录

```
cp -f /media/nfs/audio/*.wav ./ # 拷贝 /media/nfs/audio 目录下所有 wav 文件到当前目录；如若当前目
    录有同名文件则直接覆盖
cp -rP /media/datum/peacock ./ # 拷贝 /media/datum/peacock 目录到当前目录，不跟随符号链接（-P 参数
    指定）
```

第 4 章 安全相关

4.1 ssh 基本用法

ssh（Secure Shell）是一种加密的网络协议，可以在不安全的网络中安全地登录远程主机。它在 Linux/Unix 系统上被广泛使用，用于远程登录和文件传输。它可以替代 telnet 和 ftp，提供更安全的连接。它也可以用于实现两台主机之间的安全通信。它的优势在于它可以使用公钥加密，确保数据传输过程中不被窃取或篡改。

ssh 工具有不同的实现，常见的是 OpenSSH .

基本用法: ssh 参数选项 user@hostname [command]

常用选项

★ *-A* 开启 ssh agent 转发功能
★ *-a* 关闭 ssh agent 转发功能
★ *-D* 设定动态端口转发 port
★ *-F* 设定配置文件
★ *-f* 后台执行 ssh
★ *-i* 指定本次 ssh 执行使用的私钥文件
★ *-J* 设定一个跳转服务器
★ *-L* 设定本地端口转发
★ *-N* 本次 ssh 在远端不执行任何命令, 在端口转发时建议打开这个选项
★ *-n* 将标准输入重定向到 /dev/null, 和 -f 配合使用
★ *-o* 设定 ssh 参数
★ *-p* 设定 ssh 使用 tcp 端口
★ *-q* 安静模式
★ *-R* 设定远程端口转发
★ *-T* 不分配伪终端
★ *-t* 强势分配伪终端
★ *-v* 输出冗余信息
★ *-X* 开启 X11 转发
★ *-x* 关闭 X11 转发

常见用法和示例

1. 用户名和密码登录, 以 cloudnie 身份登录 host 172.29.40.153

```
# ssh cloudnie@172.29.40.153
cloudnie@172.29.40.153's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-58-generic x86_64)
```

2. 使用 key 登录

   (a). 如果没有 ssh key，可以通过 ssh-keygen 生成一对公钥和私钥

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/cloud/.ssh/id_rsa): cloudnie_key
```

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in cloudnie_key.
Your public key has been saved in cloudnie_key.pub.
The key fingerprint is:
SHA256:AWbvmXOR6aj/sxHfyWSTdx0OrFU9R5QoUQI6yRpg9mc cloud@NJT.local
The key's randomart image is:
+---[RSA 3072]----+
|   +  +  ..oo..+=|
|  o oo.oo  +o..oo|
|     o Eo + .+ .o|
|      =..* .o o.o|
|     .  S +.  =.+|
|       . o o = +.|
|      .   . . +  |
|       .  ..     |
|        ..oo     |
+----[SHA256]-----+
# ls cloudnie_key*
cloudnie_key    cloudnie_key.pub
```

(b). 通过 ssh-add 将新生成的私钥加入到 OpenSSH agent 中:

```
# ssh-add cloudnie_key
Identity added: cloudnie_key (cloud@NJT.local)
```

(c). 将公钥添加到服务器信任列表, 有两种方法

第一种方法是使用 ssh-copy-id:

```
# ssh-copy-id -i cloudnie_key cloudnie@172.29.40.153
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "cloudnie_key.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that
     are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is
     to install the new keys
cloudnie@172.29.40.153's password:

Number of key(s) added:        1

Now try logging into the machine, with:   "ssh 'cloudnie@172.29.40.153'"
and check to make sure that only the key(s) you wanted were added.
```

第二种方法是直接编辑服务器上的 authorized_keys 文件，将公钥的内容直接追加到文件末尾:

```
# ssh cloudnie@172.29.40.153 "cat >> /home/cloudnie/.ssh/authorized_keys" < cloudnie_key.pub
cloudnie@172.29.40.153's password:
```

(d). 登录远程服务器

```
# ssh cloudnie@172.29.40.153
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-58-generic x86_64)
```

3. ssh 登录后默认是执行 shell，但可以指定其他命令来代替 shell 执行, 甚至可以不执行任何命令
   (a). 登录后查看系统时间

```
# ssh cloudnie@172.29.40.153 date
2023年 03月 19日 星期日 17:12:02 CST
```

   (b). 发送本地文件到远程服务器, 比如将本地.ssh/config 上传到远程服务器 (scp 也可以做到)

```
# ssh cloudnie@172.29.40.153 "cat >> /home/cloudnie/.ssh/config" < ~/.ssh/config
```

   (c). 从远程服务器接收文件 /home/cloudnie/.ssh/authorized_keys, 本地存储为 tmp.txt (scp 也可以做到)

```
# ssh cloudnie@172.29.40.153 cat /home/cloudnie/.ssh/authorized_keys > tmp.txt
```

   (d). ssh 到远程服务器，但不执行任何命令，包括 shell

```
# ssh -N cloudnie@172.29.40.153
```

4. 使能 ssh agent 验证转发功能 (-A) 示例 (前置提交：参考上面的例子，将受信任的公钥通过 ssh-copy-id 放到服务器 172.29.40.153 上)
   (a). 开启验证转发 (-A), 从本地远程到 172.29.40.153

```
# ssh -A cloudnie@172.29.40.153
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-58-generic x86_64)
```

   从 172.29.40.153 远程登录 172.29.40.174, 因为开启了 ssh agent 转发，不需要输入密码

```
# ssh cloudnie@172.29.40.174
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-153-generic x86_64)
```

   (b). 关闭验证转发 (-a), 再次从本地远程到 172.29.40.153

```
# ssh -a cloudnie@172.29.40.153
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.15.0-58-generic x86_64)
```

   从 172.29.40.153 远程登录 172.29.40.174, 因为关闭了 ssh agent 转发，需要输入密码

```
# ssh cloudnie@172.29.40.174
cloudnie@172.29.40.174's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-153-generic x86_64)
```

   测试验证转发功能，可以打开 -v 选项，从 log 中可以清楚看到使用了本地存储的私钥
5. 中间服务器跳转 (-J), 适用于服务器无法直接访问，需要一个中间服务器跳转；比如本地工作机无法 ssh 登录服务器 172.29.102.188, 但 172.29.40.153 可以。那我们可以让 172.29.40.153 作为跳转服务器。

```
# ssh -J cloudnie@172.29.40.153 tony_nie@172.29.102.188
```

   当然，先 ssh 到 172.29.40.153, 再 ssh 到 172.29.102.188 也可以达到同样目录。另外，本地端口转发也可以达到同样目的, 只是没有上述命令简洁明了:
6. 在配置文件中指定 HOST 172.29.40.153 的端口号为 1432 和用户名 cloudnie, 私钥 cloudnie_key; 通过修改 $HOME.ssh/config 文件, 追加以下内容就可以达到上述目的.

```
Host 172.29.40.153
    HostName 172.29.40.153
    IdentityFile /home/cloudnie/.ssh/cloudnie_key
    Port 29418
    User cloudnie
```

4.2 SSH 端口转发

SSH 端口转发有两大作用, 其一是让内部不可见的 service 对外可见; 其二是提供一个加密的通道, 让数据变的安全.

在介绍端口转发之前, 让我们先对一些概念达成共识. SSH 协议本身是 CS 架构, 以 open-ssh 为例, 提供了一个 client 端工具 ssh, 和 server 端工具 sshd (通常作为常住进程 Daemon). 下文中, 相关术语描述如下:

- SSH 泛指协议
- ssh 是指 SSH 协议的 client 端工具
- SSH Client 是指 ssh 进程
- SSH Daemon 是指 SSH Server 的 Daemon 进程.

很多同学觉得 SSH 端口转发难以理解, 使用时往往一头雾水, 理不清 -L 和 -R 的区别. 哈哈, 其实这就是笔者刚开始的真实写照. 换个角度来理解问题, 如果我们能搞清楚 -L 和 -R 的使用场景, 或许它们的区别就一目了然了.

先补充一下什么是端口转发. 端口转发是指将源端口 (TCP 或 UDP) 上收到的数据转发给目的端口. 当然目的端口可以是本机的, 也可以是其他机器的. 不过 SSH 的端口转发通常是 TCP 协议上的.

SSH 支持三种端口转发, 分别通过三个参数 -L, -R 和 -D 来指定. 很多网站或博主将这这三种端口转发分别翻译为本地端口转发, 远程端口转发和动态端口转发. 笔者觉得翻译得非常贴切, 我们就绪沿用这种叫法. 前期我们对这种叫法或许会有些疑惑, 后面再解释.

本地端口转发 -L

本地端口转发是 SSH Client 监听端口, SSH Daemon 负责和 Service 对接. 使用场景如图 4.1 所示.

1. 内网主机 SERVICE 提供了一种稳定的 service
2. 主机 S 可以访问这个 service, 并且主机 S 本身开启了 SSH Daemon
3. 主机 X 无法直接访问内网主机 SERVICE
4. 主机 X 可以 ssh 到主机 S (主机 S 运行了 ssh daemon), 建立转发隧道 (tunnel)
5. 主机 X 或其他主机可以通过 tunnel 访问内网 service

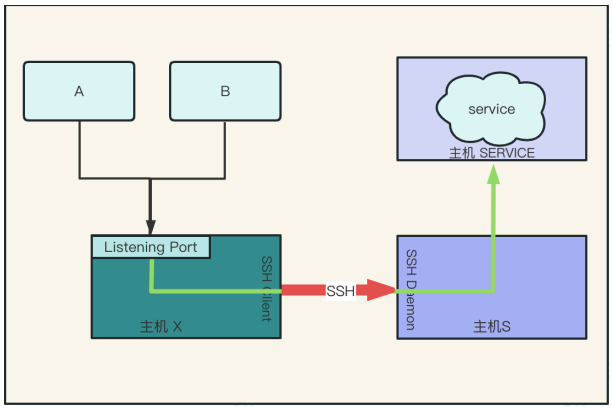
让我们看一个实际的例子:

- 内网主机 SERVICE(172.29.102.188) 在 8080 端口提供了 http service
- 主机 S(172.29.40.174) 可以访问内网的 HTPP service
- 主机 X(172.29.40.153) 无法访问 HTTP service
- 主机 X 可以 ssh 到主机 S

通过在主机 X上执行下面这条命令, 我们将主机 X 的 51432 端口映射到内网主机 SERVICE 的 8080 端口; 主机 X 就可以通过自己的端口 51432 来访问内网的 HTTP service 了.

```
#ssh  -NfL 51432:172.29.102.188:8080 cloudnie@172.29.40.174
```

图 4.1: 本地端口转发使用场景



如果主机 A, B 等也想访问上述 service 该如何处理? 很简单, 主机 X 在做端口映射时, 加上参数 -g, 其他主机就可以复用这条端口映射路径. 既访问主机 X 的 51432 端口, 来访问 HTTP service.

```
#ssh  -gNfL 51432:172.29.102.188:8080 cloudnie@172.29.40.174
```

本地端口转发基本用法: ssh -L [bind address]:<LP>:<Service IP>:<SP> <S>

参数解释如下:

★ *bind address*, 本地主机地址, 可以不给, 默认监听全部

★ *LP*, 本地端口

★ *Service IP*, Service IP 地址

★ *SP*, 远程 Service 端口

★ *S*, SSH Server

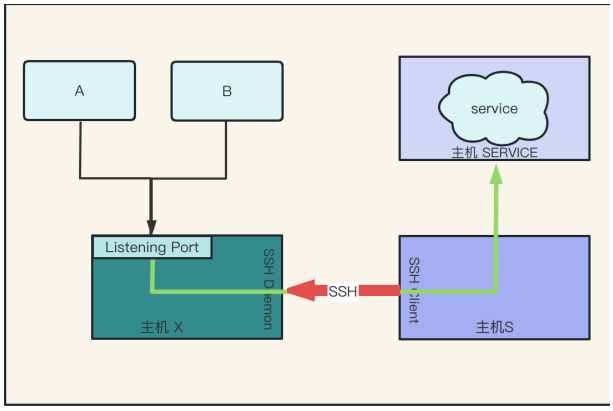
注意, 主机 SERVICE 主机和主机 S 可以是同一台主机.

远程端口转发 -R

远程端口转发是 SSH Deamon 监听端口, SSH Client 负责和 Service 对接; 使用场景如 4.2 所示, 描述如下:

1. 内网主机 SERVICE 提供了一种稳定的 service
2. 主机 S 可以访问这个 service
3. 主机 X 无法直接访问内网主机 SERVICE, 也不能直接 SSH 到主机 S
4. 主机 S 可以 ssh 到主机 X (主机 X 运行了 ssh daemon), 建立转发隧道 (tunnel)
5. 主机 X 或其他主机可以通过 tunnel 访问内网 service

图 4.2: 远程端口转发使用场景



和本地端口转发最大的差异在于 *主机 X 不能直接 SSH 到主机 S, 但主机 S 可以 SSH 到主机 X.*

让我们看一个实际的例子:

- 内网主机 SERVICE(172.29.102.188) 在 8080 端口提供了 http service
- 主机 S(172.29.40.174) 可以访问内网的 HTPP service
- 主机 X(172.29.40.153) 无法访问 HTTP service
- 主机 S 可以 ssh 到主机 X

通过在主机 S 上执行下面这条命令, 我们将主机 X 的 51432 端口映射到内网主机 SERVICE 的 8080 端口; 主机 X 就可以通过自己的端口 51432 来访问 SERVICE 的 HTTP service 了.

```
#ssh  -NfR 51432:127.0.0.1:172.29.102.188:8080 cloudnie@172.29.40.153
```

同理, 加上参数 -g, 其他主机就可以复用这条端口映射路径. 既访问主机 X 的 51432 端口, 就可以访问主机 SERVICE 的 HTPP 服务.


```
#ssh  -NfR 51432:127.0.0.1:172.29.102.188:8080 cloudnie@172.29.40.153
```


这里有一个地方需要注意, 通常来说主机 X 是可见的, 主机 S 不可见; 所以需要一个特殊的途径, 方便我们在主机 S 上执行 ssh 命令.

远程端口转发基本用法: ssh -R [bind address]:<RP>:<Service IP>:<SP> <S>

参数解释如下:

★ *bind address*, 指定使用远程主机的哪个 IP 地址 (如果有多个的话), 可以不给, 默认监听全部

★ *RP*, 远程端口

★ *Service IP*, Service IP 地址

★ *SP*, 远程 Service 端口

★ *S*, SSH Server

总结

让我们总结一下本地端口转发和远程端口转发:

- 本地和 远程两个词是相对于 SSH Client 来讲的, 因为端口转发的命令中, 我们启动的都是 SSH Client
- 本地端口转发是 ssh client 进程自身监听 TCP/IP 端口, 所以是本地端口;
- 远程端口转发是 ssh client 请求 ssh daemon 进程监听 TCP/IP 端口, 这个端口通常是在另外一台主机 (ssh daemon 所在), 所以是远程端口.
- 选择本地端口转发还是远程端口转发, 在于外网主机 X 能充当什么角色; 如果 X 可以作为 SSH Client, 就是本地端口转发; 如果 X 必须作为 SSH Daemon, 就是远程端口转发

动态端口转发 -D

动态端口转发的目的是让 ssh 进程成为 SOCKS4 或者 SOCKS5 服务进程.

```
ssh -TfnND 8089 cloudnie@cloudnie.com
```

4.3 scp

scp 用来在不同主机之间以加密的方式进行文件拷贝, 本身是基于 ssh 实现的. 参考 ssh 基本用法, 通过 ssh 命令也也可以实现文件的拷贝功能.

基本用法: scp [346BCpqrTv] [-P port] [[user@]host1:]file1

常用参数说明:

★ *-P* 设定 SSH Daemon 使用端口号
★ *-r* 递归拷贝目录
★ *-i* 指定使用的私钥
★ *-v* 打开 verbose log

常见用法和示例

1. 将本地文件拷贝到远程主机

```
scp -i ~/.ssh/tony_nie_np pic2.bgr cloudnie@172.29.40.153:~/
pic2.bgr                                              100\% 3072KB  11.1MB/s   00:00
```

2. 从远程主机获取文件

```
scp cloudnie@172.29.40.153:~/pic2.bgr .
pic2.bgr                                      100\% 3072KB  11.2MB/s   00:00
```

3. 将本地目录拷贝到远程主机

```
scp -P 22   -r ./data cloudnie@172.29.40.153:~/
lower.txt                                             100\%   29    30.1KB/s   00:00
list.txt                                              100\%   73    67.8KB/s   00:00
time.txt                                              100\%   29    31.4KB/s   00:00
pic2.bgr                                              100\% 3072KB  11.1MB/s   00:00
upper.txt                                             100\%   29    43.9KB/s   00:00
pic.bgr                                               100\% 3072KB  11.1MB/s   00:00
```

4.4 rsync

rsync 是一个高性能且用途广泛的文件和目录拷贝工具, 广泛应用于 Unix/Linux, Mac, Windows 等操作系统中; 它不仅可以拷贝本地文件, 也可以在不同主机之间拷贝文件. rsync 因为 delta-transfer 算法而出名, 它能够快速地比较源和目标文件或者目录的内容，并只复制有变化的文件或者目录，从而实现快速的文件传输和复制.

基本用法: rsync [选项列表] SOURCE DEST

常用选项:

★ *-v* 打开 verbose log
★ *-q* 安静模式, 减少 log 输出
★ *-c* 通过计算 checksum 判断文件是否需要传输, 否则是根据文件大小和修改时间
★ *-r* 递归拷贝目录
★ *-R* 位置目录组织结构
★ *-l* 遇到符号连接时, 创建符号链接
★ *-L* 遇到符号链接是, 拷贝其指向的内容

常用示例

1. 将本地目录推送到远程主机

```
# rsync -rv img cloudnie@172.29.40.153:~/
cloudnie@172.29.40.153's password:
sending incremental file list
img/
img/loopfile.img
img/mem.img
img/part2.img
img/ramdon.img
img/sda_1st.img

sent 26,221,705 bytes  received 115 bytes  5,827,071.11 bytes/sec
total size is 26,214,912  speedup is 1.00
```

2. 从远程主机获取目录

```
# rsync -rv cloudnie@172.29.40.153:~/data .
cloudnie@172.29.40.153's password:
receiving incremental file list
data/
data/a.jpg
data/all.txt
data/list.txt
data/lower.txt
data/pic.bgr
data/pic2.bgr
data/time.txt
data/tmp.txt
data/upper.txt

sent 199 bytes  received 6,294,103 bytes  2,517,720.80 bytes/sec
total size is 6,291,944  speedup is 1.00
```

3. 本地目录拷贝

```
# rsync -rv ./data /media/datum/
sending incremental file list
data/
data/a.jpg
data/all.txt
data/list.txt
data/lower.txt
data/pic.bgr
data/pic2.bgr
data/time.txt
data/tmp.txt
data/upper.txt

sent 6,294,087 bytes received 191 bytes 12,588,556.00 bytes/sec
```

```
total size is 6,291,944 speedup is 1.00
```

4.5 md5sum/sha256sum/sha512sum

md5sum, sha256sum 以及 sha512sum 可以用来计算和比较文件的消息摘要, 从而确保文件的完整性和防止篡改. 针对一些安全级别比较高的场景, 拿到文件之后, 可以在本地计算消息摘要, 和官方公布的消息摘要做对比, 确认拿到的文件是完整的, 以及没有被篡改过. 更加安全的方式是使用数字签名. 因为计算机性能的逐渐提高, md5sum 开始变的不安全; 所以大家逐渐开始使用 sha256sum 或 sha512sum 代替 md5sum.

很多软件在提供下载链接的同时, 会在网站上公布对应的消息摘要, 方便下载之后做检查, 比如大名鼎鼎的 FTP 软件 FileZilla. 参考 4.3

图 4.3: fileZilla 下载页面

常见示例

1. 计算文件的 sha512 摘要

```
# sha512sum FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2
35a8fa68cc9754db4051d58b1f77c12c2b291e240b2f2943f2c16d6fdf1208300215470165e3add290383f712a1f191db
3db778d1302276d41180ef64727f7a4 FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2
```

2. 计算文件的 sha256 摘要

```
# sha256sum FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2
54e14c35d36fb60a9949764c83c70e138f219249ec12810b14c492203966d849 FileZilla_3.63.2.1_x86_64-linux
-gnu.tar.bz2
```

3. 计算文件的 md5 摘要

```
# md5sum FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2
0819c6079519ce485275cd527ee3aa36 FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2
```

4. 比较文件的 sha256 摘要, 确认文件是完整和正确的

```
sha512sum -c FileZilla.sha512sum
FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2: OK
```

FileZilla.sha512sum 的内容为:

```
# cat FileZilla.sha512sum
35a8fa68cc9754db4051d58b1f77c12c2b291e240b2f2943f2c16d6fdf1208300215470165e3add290383f712a1f191db
3db778d1302276d41180ef64727f7a4 FileZilla_3.63.2.1_x86_64-linux-gnu.tar.bz2
```

结束语

下个版本, 还会增加以下工具

1. 字符串处理相关: awk/sed/grep/hexdump/xxd
2. 多媒体相关: ffmpeg/imagemagick
3. 网络相关:curl/wget

啰啰嗦嗦说了这么多，让我们相见于江湖吧。

愿祖国繁荣昌盛，人民安居乐业.