【1-4】linux文件和文件夹操作-进阶

笔记本： 备课_linux
创建时间： 2022/4/16 14:04 更新时间： 2022/4/26 17:19
作者： 兰鸣人花道
URL： https://www.runoob.com/note/29134

1、创建文件

- vi
- touch
    - 文件不存在，创建新的
    - 文件存在，会修改文件的修改时间
- 都可以使用绝对路径和相对路径
- rm -f d1/* # 删除d1文件夹下的所有文件，但是保留d1，就是d1目录不会被删除
- rm -rf d1/* # 删除d1文件夹下的所有文件，但是保留d1，就是d1目录不会被删除
- rm -rf d1/ # 删除包括d1目录在内的文件夹下的所有文件

2、查看文件内容

- cat
    - -n，显示行号，number
    - cat -n /etc/passwd
- more
    - enter，一行一行的往下翻页
    - space[空格]，一页一页的翻，ctrl+F
    - q退出 (man也是退出)
- less
    - k，后退，一行一行的 (vi)
    - j，前进，一行一行的 (vi)
    - space[空格键]
    - enter
    - page up
    - page down
    - ctrl + B，往文件开头方向翻屏
    - ctrl + f，跟more命令一样，往文件结尾方向翻屏
    - 方向键向上、向下
    - /要搜索的字符串，n，N继续搜索
    - ？要搜索的字符串， N，n继续搜索，方向跟/搜索的方向相反
    - q，退出
- head
    - head 文件名 # 默认展示文件前10行内容
    - head -n 数字 文件名 # 展示你给定数字的行
    - head -数字 文件名 # 同上

```
[root@localhost ~]# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin

[root@localhost ~]# more /etc/passwd

[root@localhost ~]# less /etc/passwd

[root@localhost ~]# head -5 functions
# -*-Shell-script-*-
#
# functions This file contains functions to be used by most or all
#       shell scripts in the /etc/init.d directory.
#
[root@localhost ~]#
```

```
[root@localhost ~]# head -n 2 functions
# -*-Shell-script-*-
#
[root@localhost ~]# head -n 5 functions
# -*-Shell-script-*-
#
# functions This file contains functions to be used by most or all
#       shell scripts in the /etc/init.d directory.
#
[root@localhost ~]#
```

- tail
    - 查看日志尾部文件
    - tail 文件
        - 默认也是10行，但是是尾部的
    - tail -n 5 文件
    - tail -数字 文件
    - tail -f 文件，实时监控文件动态变化
    - tail -f /var/log/messages
    - 查看访问nginx服务器日志
        - tail -f /var/log/nginx/access.log
        - tail -f /var/log/nginx/error.log
        - 怎么退出，ctrl + c

```
[root@localhost ~]# tail -n 2 /etc/passwd
chrony:x:998:996::/var/lib/chrony:/sbin/nologin
tom:x:1000:1000::/home/tom:/bin/bash

[root@localhost ~]# tail -3 /etc/passwd
postfix:x:89:89::/var/spool/postfix:/sbin/nologin
chrony:x:998:996::/var/lib/chrony:/sbin/nologin
tom:x:1000:1000::/home/tom:/bin/bash
[root@localhost ~]#



#    监控nginx日志

========这里学生可以不用理解========
[root@localhost nginx_log]# docker run -d -p 80:80 --name m1 -v /root/nginx_log/:/var/log/nginx/
nginx:1.17.1

[root@localhost nginx_log]# docker ps
CONTAINER
ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                NAMES
8831f90e395c        nginx:1.17.1        "nginx -g 'daemon ..."   2 minutes ago       Up 2
minutes        0.0.0.0:80->80/tcp   m1
[root@localhost nginx_log]#
========这里学生可以不用理解========

[root@localhost nginx_log]# ls
access.log  error.log
[root@localhost nginx_log]# pwd
/root/nginx_log

[root@localhost nginx_log]# tail -f access.log
172.17.0.1 - - [20/Apr/2022:13:59:47 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"
172.17.0.1 - - [20/Apr/2022:14:00:29 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"
192.168.100.157 - - [20/Apr/2022:14:00:50 +0000] "GET / HTTP/1.1" 200 612 "-" "Mozilla/5.0 (Windows NT
10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.88 Safari/537.36" "-"
192.168.100.157 - - [20/Apr/2022:14:00:59 +0000] "GET /xxxx HTTP/1.1" 404 555 "-" "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/100.0.4896.88 Safari/537.36" "-"
```

3、通配符

- *，可以任意多个
    - 注意，都是区分大小写
    - ls test*   指定了显示test开头的文件或者目录
    - ls *test*   指定了显示只要包含了test字符串的文件或者目录，
    - ls *test*   不会显示Test带大写字母这种
    - ls /etc/*release   指定/etc/目录下，以release结尾的文件
    - ls *.log   不知道文件名的时候，查找以log结尾的文件(日志文件)
- ?，表示一个字符
    - 必须有，必须代表一个字符在那里
    - ls test?.sh 指定以test开头，接一个字符，后面跟.sh结尾的文件

- []，
  - [346abc] 代表3,4,6，a,b,c　都可以匹配到
  - [2-5]　代表2到5，即2,3,4,5，都可以匹配到
  - [a-z]　代表所有的英文字母，不区分大小写
  - [a-f]，表示匹配a，b，c，d，e，f，大写也可以
  - [af]，表示匹配a或者f，大写也可以
- mkdir -pv dir/file{1,2,3}
  - 表示创建dir目录及子目录file1，file2，file3
  - 就是一个简写的语法

4、复制

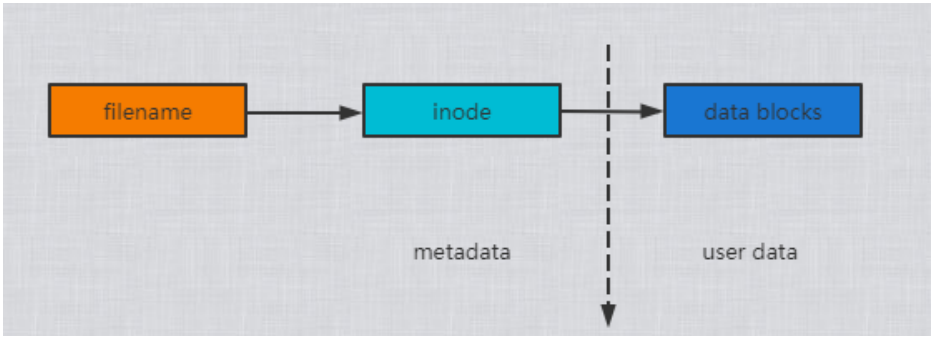
- cp
  - copy的缩写
  - cp 源文件 目标文件名
    - cp /etc/init.d/functions .　　复制/etc/init.d/目录下的functions这个文件到 .（当前目录），名称我们没有处理
    - cp /etc/init.d/functions ./　　同上，(.)换成了(./)
    - cp functions /root/　　复制的时候，进入了源文件的目录，就可以用相对路径，再复制到/root/目录下
    - cp /etc/passwd mypasswd　　复制/etc/目录下的passwd文件，到当前目录，并重命名位mypasswd
  - -r，-R
    - cp -r nginx_log/ /tmp/　　将nginx_log/这个目录复制到/tmp/下
      - 如果不加-r，报错 cp: omitting directory ‘nginx_log/’
    - cp access.log error.log /tmp/aaa　　如果源文件(中间的就是源文件) 有多个，最后一个你要拷贝的位置，只能是目录
      - cp: target ‘/tmp/aaa’ is not a directory
  - -i，--interactive
    - 交互式提示，prompt before overwrite (overrides a previous -n option)
    - cp，是一个别名　alias cp='cp -i'，就相当于我们现在用cp ，其实用的是cp -i
- 复制目录
  - cp -r nginx_log/ /tmp/　　将nginx_log/这个目录复制到/tmp/下

5、移动和重命名

- mv
  - mv 源文件 目标文件　　# 表示重命名源文件为新的名字
    - mv README readme
  - mv 目录1 目录2 # 结合上下文，看是移动，还是说改名
    - mv test1/ test2/　#如果test2不存在，那么就是改名；如果test2存在，就是移动
    - mv 文件/目录1 文件/目录2　...目标文件夹　　#最后一个一定是文件夹，　表示移动
  - mv -i　# 移动到的目录里面，有重名文件，需要手动确认
- -i参数总结：
  - -i参数，交互式提示，在rm，cp，mv里面都有
  - 目的是为了安全，尤其是系统重要的配置文件，在覆盖或者删除前要手动确认，避免出现一不小心就干掉正常的配置文件的情况，避免误操作使系统挂掉

6、硬链接和软链接

- 为什么需要链接？
  - 实现文件的共享使用，比如那个路径有其他程序在引用，不能随便移动
  - 提供了隐藏文件路径、增加权限安全以及节省存储等好处
  - 链接与复制的区别：链接其实就是一个指向，不是复制的文件内容，所以节省磁盘空间
- 文件链接：
  - 我们说文件都有文件名和数据，在linux上分为两个部分，元数据(metadata)及用户数据(user data)，也叫做数据块，data block
  - 用户数据，即文件数据块(data block)，数据块是记录文件真实内容的地方
  - 元数据，即metadata，记录的是文件的附加属性，比如文件大小，创建时间，所有者等信息
  - 在linux中，元数据中的inode号才是文件的唯一标识，而非文件名
    - inode，是文件元数据的一部分，但是它并不包含文件名，inode就是索引节点号
  - 文件名只是为了方便人们记忆和使用，而系统或者程序则是通过inode来寻找文件数据块的，系统这种算法可以快速定位到要想查找的文件

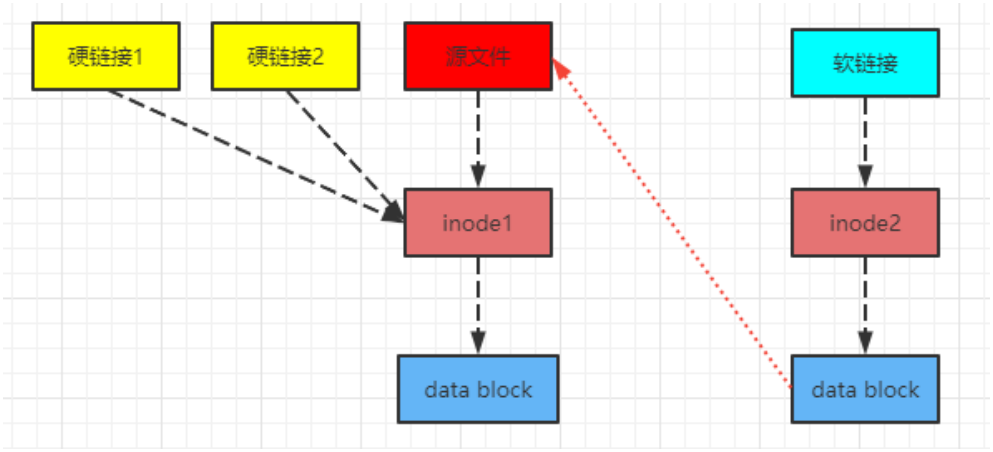


- 查看文件的inode
  - ls -li
  - stat 文件名

```
[root@lanhai link_test]# ls -i file01
35052575 file01
[root@lanhai link_test]#

[root@lanhai link_test]# stat file01
  File: ‘file01’
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: fd00h/64768d    Inode: 35052575    Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Context: unconfined_u:object_r:admin_home_t:s0
Access: 2020-03-01 12:05:53.989000000 +0800
Modify: 2020-03-01 12:05:53.989000000 +0800
Change: 2020-03-01 12:06:03.189000000 +0800
Birth: -
[root@lanhai link_test]#
```

- 链接的种类：
  - 硬链接 hard link
    - ln 源文件 硬链接文件名
  - 软链接 soft link 或者symbolic link
    - ln -s 源文件 软链接文件名
- 硬链接：
  - 一个inode可以对应多个文件名，则这些文件称为硬链接
  - 换言之，硬链接就是同一个文件使用了多个别名
  - 特点
    - 文件有相同的inode及data block
    - 只能对已存在的文件进行创建
    - 不能跨文件系统进行硬链接的创建
    - 不能对目录进行创建，只能对文件进行创建硬链接
    - 删除一个硬链接文件不会影响其他有相同inode号的文件
- 软链接：
  - 软链接也叫做符号链接，symbolic link
  - ls -lt里面权限位的第一个位表示为l(L的小写)
    - 如果第一个位是-，表示普通文件
    - 如果第一个位是d，表示目录
    - 如果第一个位是l，表示是软连接文件
    - 如果b，块设备文件
    - s，socket文件
    - c，字符设备文件
  - 若文件的用户数据块中存放的内容是另一文件的路径名的指向，那么这个文件就是软链接
  - 软链接就是一个普通文件，只是数据块内容有点特殊
  - 类似windows里面的快捷方式
  - 特点
    - 软链接有自己的文件属性及权限等
    - 可以对不存在的文件或者目录创建软链接
    - 软链接可以跨文件系统
    - 软链接可以对目录进行创建，这点与硬链接不同
    - 创建软链接时，链接计数i_nlink不会增加
    - 删除软链接，不会影响被指向的文件，也就是不影响源文件
    - 对于目录来说，你进去操作是一样的，操作的是同一个目录，你如果进去删除了文件，相当于你进入原来的目录，进行操作
    - 但是如果源文件或者说被指向的文件被删除，则相关软链接被称为死链接(即dangling link)
    - 如果把这个文件恢复了(路径被重新创建)，那么这个死链接可以恢复为正常的软链接
    - 软链接的inode不同
  - 软链接用在何处？
    - 当同一个文件需要在多个位置被用到的时候，就可以使用软链接
    - 环境变量的时候，再给大家演示(yum 安装)
- 对源文件的修改，软链接、硬链接的文件内容查看的时候也是一样的修改，因为指向的是同一个文件的内容
- 创建一个文件
  - touch file01

- 分别创建硬链接、软链接
  - ln file01 file01_hardlink
  - ln -s file01 file01_softlink
- 硬链接的inode相同，软链接的inode不同
- 软、硬链接的访问

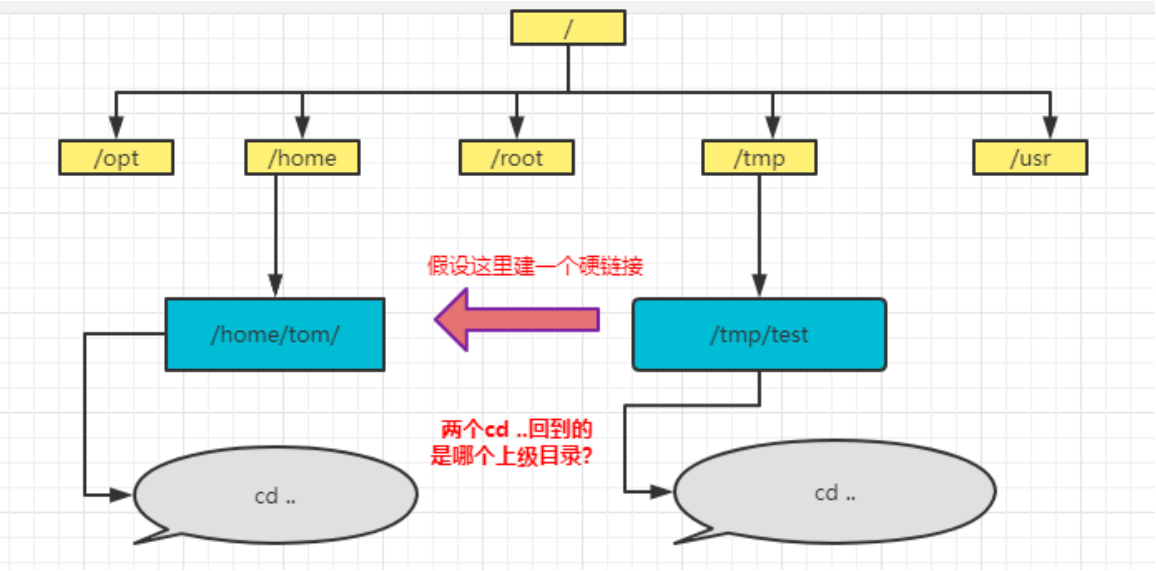


硬连接
硬连接指通过索引节点来进行连接。在 Linux 的文件系统中，保存在磁盘分区中的文件不管是什么类型都给它分配一个编号，称为索引节点号(Inode Index)。在 Linux 中，多个文件名指向同一索引节点是存在的。比如：A 是 B 的硬链接（A 和 B 都是文件名），则 A 的目录项中的 inode 节点号与 B 的目录项中的 inode 节点号相同，即一个 inode 节点对应两个不同的文件名，两个文件名指向同一个文件，A 和 B 对文件系统来说是完全平等的。删除其中任何一个都不会影响另外一个的访问。
硬连接的作用是允许一个文件拥有多个有效路径名，这样用户就可以建立硬连接到重要文件，以防止“误删”的功能。其原因如上所述，因为对应该目录的索引节点有一个以上的连接。只删除一个连接并不影响索引节点本身和其它的连接，只有当最后一个连接被删除后，文件的数据块及目录的连接才会被释放。也就是说，文件真正删除的条件是与之相关的所有硬连接文件均被删除。

软连接
另外一种连接称之为符号连接（Symbolic Link），也叫软连接。软链接文件有类似于 Windows 的快捷方式。它实际上是一个特殊的文件。在符号连接中，文件实际上是一个文本文件，其中包含的有另一文件的位置信息。比如：A 是 B 的软链接（A 和 B 都是文件名），A 的目录项中的 inode 节点号与 B 的目录项中的 inode 节点号不相同，A 和 B 指向的是两个不同的 inode，继而指向两块不同的数据块。但是 A 的数据块中存放的只是 B 的路径名（可以根据这个找到 B 的目录项）。A 和 B 之间是“主从”关系，如果 B 被删除了，A 仍然存在（因为两个是不同的文件），但指向的是一个无效的链接。

```
#    创建文件，硬链接，软链接
#    查看各个文件的inode

[root@lanhai link_test]# touch file01
[root@lanhai link_test]# ln file01 file01_hardlink
[root@lanhai link_test]# ln -s file01 file01_softlink
[root@lanhai link_test]# ls -li
total 0
35052575 -rw-r--r--. 2 root root 0 Mar  1 12:05 file01
35052575 -rw-r--r--. 2 root root 0 Mar  1 12:05 file01_hardlink
35052576 lrwxrwxrwx. 1 root root 6 Mar  1 12:06 file01_softlink -> file01

#    不能对不存在的文件创建硬链接，要报错
[root@localhost link_dir]# ln xxxx xxxx_hard_link
ln: failed to access ‘xxxx’: No such file or directory
[root@localhost link_dir]#

#    不能对目录创建硬链接
[root@localhost link_dir]# ln dir1 dir1_hlink
ln: ‘dir1’: hard link not allowed for directory
[root@localhost link_dir]#

#    对不存在的文件创建软链接后
17411146 lrwxrwxrwx. 1 root root     7 Apr 26 17:04 xx_soft_link -> xxxxxxx    #这个在闪
[root@localhost link_dir]# cat xx_soft_link
cat: xx_soft_link: No such file or directory
[root@localhost link_dir]#
```

- 拓展？为什么不能对目录创建硬链接？



7、课堂练习

1)、在/home目录下创建两个子目录dir01和dir02
2)、在/home/dir01目录下创建一个文件file01
3)、把上一步的file01另存为file01.bak，保存在/home/dir02目录下
4)、在/home/dir01目录下为上一步的file01.bak文件创建一个软链接文件file01.bak.sl
5)、移动上一步的file01.bak.sl文件到/home/dir02目录下
6)、分别列出/home/dir01和/home/dir02目录下所有的文件，截图发到群里

8、课后作业

1）怎样查看文件scp.log的末尾20行？
2）怎样查看文件scp.log的前5行？
3）怎样查看当前目录以“.log”结尾的文件或目录？
4）当前目录为/var ，怎么把文件/var/log/scp.log 移动到/var/log/backup下？请使用相对路径
5）怎么删除/var/log 下所有的以.log 结尾的文件？
6）怎么把/var/log目录以及该目录下的所有文件拷贝到/home/jay目录？
7）分别为文件/var/log/scp.log在当前目录下创建一个软链接scp_soft.log和硬链接scp_hard.log，写出创建链接的命令？如果我删除了/var/log/scp.log文件，这两个链接文件还能查看其内容吗？
8）硬链接与软链接的区别？
