Linux

2017年8月21日 10:07

1. **硬链接和软链接**

清单 1. Linux 系统的顶层目录结构

```
/           根目录
├─── bin    存放用户二进制文件
├─── boot   存放内核引导配置文件
├─── dev    存放设备文件
├─── etc    存放系统配置文件
├─── home   用户主目录
├─── lib    动态共享库
├─── lost+found          文件系统恢复时的恢复文件
├─── media  可卸载存储介质挂载点
├─── mnt    文件系统临时挂载点
├─── opt    附加的应用程序包
├─── proc   系统内存的映射目录，提供内核与进程信息
├─── root   root 用户主目录
├─── sbin   存放系统二进制文件
├─── srv    存放服务相关数据
├─── sys    sys 虚拟文件系统挂载点
├─── tmp    存放临时文件
├─── usr    存放用户应用程序
└─── var    存放邮件、系统日志等变化文件
```

Linux 与其他类 UNIX 系统一样并不区分文件与目录：目录是记录了其他文件名的文件。使用命令 mkdir 创建目录时，若期望创建的目录的名称与现有的文件名（或目录名）重复，则会创建失败。

```
# ls -F /usr/bin/zi*
/usr/bin/zip*       /usr/bin/zipgrep*  /usr/bin/zipnote*
/usr/bin/zipcloak*  /usr/bin/zipinfo*  /usr/bin/zipsplit*
# mkdir -p /usr/bin/zip
mkdir: cannot create directory `/usr/bin/zip': File exists
```

Linux 将设备当做文件进行处理，清单 2.展示了如何打开设备文件 /dev/input/event5 并读取文件内容。文件 event5 表示一种输入设备，其可能是鼠标或键盘等。查看文件 /proc/bus/input/devices 可知 event5 对应设备的类型。设备文件 /dev/input/event5 使用 read() 以字符流的方式被读取。结构体 input_event 被定义在内核头文件 linux/input.h 中。

清单 2. 打开并读取设备文件

```
int fd;
struct input_event ie;
fd = open("/dev/input/event5", O_RDONLY);
read(fd, &ie, sizeof(struct input_event));
printf("type = %d  code = %d  value = %d\n",
                                ie.type, ie.code, ie.value);
close(fd);
```

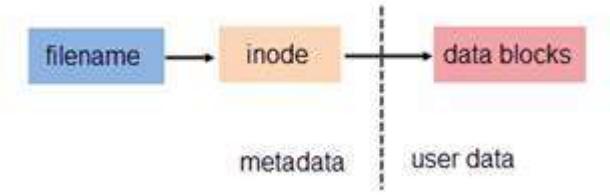
回页首

硬链接与软链接的联系与区别

我们知道文件都有文件名与数据，这在 Linux 上被分成两个部分：用户数据 (user data) 与元数据 (metadata)。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；而元数据则是文件的附加属性，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（inode 是文件元数据的一部分但其并不包含文件名，inode 号即索引节点号）才是文件的唯一标识而非文件名。文件名仅是为了方便人们的记忆和使用，系统或程序通过 inode 号寻找正确的文件数据块。图 1.展示了程序通过文件名获取文件内容的过程。

图 1. 通过文件名打开文件



清单 3. 移动或重命名文件

```
# stat /home/harris/source/glibc-2.16.0.tar.xz
 File: `/home/harris/source/glibc-2.16.0.tar.xz'
 Size: 9990512              Blocks: 19520      IO Block: 4096   regular file
Device: 807h/2055d          Inode: 2485677     Links: 1
Access: (0600/-rw-------)  Uid: ( 1000/  harris)   Gid: ( 1000/  harris)
...
...
# mv /home/harris/source/glibc-2.16.0.tar.xz /home/harris/Desktop/glibc.tar.xz
# ls -i -F /home/harris/Desktop/glibc.tar.xz
2485677 /home/harris/Desktop/glibc.tar.xz
```

在 Linux 系统中查看 inode 号可使用命令 stat 或 ls -i（若是 AIX 系统，则使用命令 istat）。清单 3.中使用命令 mv 移动并重命名文件 glibc-2.16.0.tar.xz，其结果不影响文件的用户数据及 inode 号，文件移动前后 inode 号均为：2485677。

为解决文件的共享使用，Linux 系统引入了两种链接：硬链接 (hard link) 与软链接（又称符号链接，即 soft link 或 symbolic link）。链接为 Linux 系统解决了文件的共享使用，还带来了隐藏文件路径、增加权限安全及节省存储等好处。若一个 inode 号对应多个文件名，则称这些文件为硬链接。换言之，硬链接就是同一个文件使用了多个别名（见 图 2.hard link 就是 file 的一个别名，他们有共同的 inode）。硬链接可由命令 link 或 ln 创建。如下是对文件 oldfile 创建硬链接。

```
link oldfile newfile
ln oldfile newfile
```

由于硬链接是有着相同 inode 号仅文件名不同的文件，因此硬链接存在以下几点特性：

- 文件有相同的 inode 及 data block；
- 只能对已存在的文件进行创建；
- 不能交叉文件系统进行硬链接的创建；
- 不能对目录进行创建，只可对文件创建；
- 删除一个硬链接文件并不影响其他有相同 inode 号的文件。

来自 <https://www.ibm.com/developerworks/cn/linux/l-cn-hardandsymb-links/>

首先，从使用的角度讲，两者没有任何区别，都与正常的文件访问方式一样，支持读写，如果是可执行文件的话也可以直接执行。

那区别在哪呢？在底层的原理上。

为了解释清楚，我们首先在自己的一个工作目录下创建一个文件，然后对这个文件进行链接的创建：

$ touch myfile && echo "This is a plain text file." > myfile

$ cat myfile

This is a plain text file.

现在我们创建了一个普通地不能再普通的文件了。然后我们对它创建一个硬链接，并查看一下当前目录：

```
$ ln myfile hard
$ ls -li
25869085 -rw-r--r--  2 unixzii  staff  27  7  8 17:39 hard
25869085 -rw-r--r--  2 unixzii  staff  27  7  8 17:39 myfile
```

在 ls 结果的最左边一列，是文件的 inode 值，你可以简单把它想成 C 语言中的指针。它指向了物理硬盘的一个区块，事实上文件系统会维护一个引用计数，只要有文件指向这个区块，它就不会从硬盘上消失。

你也看到了，这两个文件就如同一个文件一样，inode 值相同，都指向同一个区块。

然后我们修改一下刚才创建的 *hard* 链接文件：

```
$ echo "New line" >> hard
$ cat myfile
This is a plain text file.
New line
```

可以看到，这两个文件果真就是一个文件。

下面我们看看软链接（也就是符号链接）和它有什么区别。

```
$ ln -s myfile soft
$ ls -li
25869085 -rw-r--r--  2 unixzii  staff  36  7  8 17:45 hard
25869085 -rw-r--r--  2 unixzii  staff  36  7  8 17:45 myfile
25869216 lrwxr-xr-x  1 unixzii  staff   6  7  8 17:47 soft -> myfile
```

诶，你会发现，这个软链接的 inode 竟然不一样啊，并且它的文件属性上也有一个 l 的 flag，这就说明它与之前我们创建的两个文件根本不是一个类型。

下面我们试着删除 *myfile* 文件，然后分别输出软硬链接的文件内容：

```
$ rm myfile
$ cat hard
This is a plain text file.
New line
$ cat soft
cat: soft: No such file or directory
```

之前的硬链接没有丝毫地影响，因为它 inode 所指向的区块由于有一个硬链接在指向它，所以这个区块仍然有效，并且可以访问到。

然而软链接的 inode 所指向的内容实际上是保存了一个绝对路径，当用户访问这个文件时，系统会自动将其替换成其所指的文件路径，然而这个文件已经被删除了，所以自然就会显示无法找到该文件了。

为验证这一猜想，我们再向这个软链接写点东西：

```
$ echo "Something" >> soft
$ ls
hard    myfile soft
```

可以看到，刚才删除的 *myfile* 文件竟然又出现了！这就说明，当我们写入访问软链接时，系统自动将其路径替换为其所代表的绝对路径，并直接访问那个路径了。

总结

到这里我们其实可以总结一下了：

- 硬链接： 与普通文件没什么不同，inode 都指向同一个文件在硬盘中的区块
- 软链接： 保存了其代表的文件的绝对路径，是另外一种文件，在硬盘上有独立的区块，访问时替换自身路径。

作者：Cyandev
链接：http://www.jianshu.com/p/dde6a01c4094
來源：简书

2. **kill进程杀不掉的原因**

   一是进程已经成为僵死进程，当它的父进程将它回收或将它的父进程kill掉即可在ps输出看不到了；
   二是进程正处在内核状态中，Linux进程运行时分内核和用户两种状态，当进程进入内核状态后，会屏蔽所有信号，包括SIGKIL，所以这个时候kill -9也变得无效了。

3. **Kill 用法**

   1．命令格式：

   kill[参数][进程号]

   2．命令功能：

   发送指定的信号到相应进程。不指定型号将发送SIGTERM（15）终止指定进程。如果任无法终止该程序可用“-KILL” 参数，其发送的信号为SIGKILL(9)，将强制结束进程，使用ps命令或者jobs 命令可以查看进程号。root用户将影响用户的进程，非root用户只能影响自己的进程。

   3．命令参数：

   -l 信号，若果不加信号的编号参数，则使用“-l”参数会列出全部的信号名称

   -a 当处理当前进程时，不限制命令名和进程号的对应关系

   -p 指定kill 命令只打印相关进程的进程号，而不发送任何信号

   -s 指定发送信号

   -u 指定用户

   注意：

   1、kill命令可以带信号号码选项，也可以不带。如果没有信号号码，kill命令就会发出终止信号

(15)，这个信号可以被进程捕获，使得进程在退出之前可以清理并释放资源。也可以用kill向进程发送特定的信号。例如：

kill -2 123

它的效果等同于在前台运行PID为123的进程时按下Ctrl+C键。但是，普通用户只能使用不带signal参数的kill命令或最多使用-9信号。

2、kill可以带有进程ID号作为参数。当用kill向这些进程发送信号时，必须是这些进程的主人。如果试图撤销一个没有权限撤销的进程或撤销一个不存在的进程，就会得到一个错误信息。

3、可以向多个进程发信号或终止它们。

4、当kill成功地发送了信号后，shell会在屏幕上显示出进程的终止信息。有时这个信息不会马上显示，只有当按下Enter键使shell的命令提示符再次出现时，才会显示出来。

5、应注意，信号使进程强行终止，这常会带来一些副作用，如数据丢失或者终端无法恢复到正常状态。发送信号时必须小心，只有在万不得已时，才用kill信号(9)，因为进程不能首先捕获它。要撤销所有的后台作业，可以输入kill 0。因为有些在后台运行的命令会启动多个进程，跟踪并找到所有要杀掉的进程的PID是件很麻烦的事。这时，使用kill 0来终止所有由当前shell启动的进程，是个有效的方法。

HUP　1　终端断线

INT　2　中断（同 Ctrl + C）

QUIT　3　退出（同 Ctrl + \）

TERM　15　终止

KILL　9　强制终止

CONT　18　继续（与STOP相反， fg/bg命令）

STOP　19　暂停（同 Ctrl + Z）

来自 <http://www.cnblogs.com/wangcp-2014/p/5146343.html>

? 4. linux查看日志文件的方式

? 5. linux用过的命令

ls　显示文件或目录

-l　列出文件详细信息l(list)

-a　列出当前目录下所有文件及目录，包括隐藏的a(all)

mkdir　创建目录

-p　创建目录，若无父目录，则创建p(parent)

cd　切换目录

touch　创建空文件

echo	创建带有内容的文件。
cat	查看文件内容
cp	拷贝
mv	移动或重命名
rm	删除文件
-r	递归删除，可删除子目录及文件
-f	强制删除
find	在文件系统中搜索某文件
wc	统计文本中行数、字数、字符数
grep	在文本文件中查找某个字符串
rmdir	删除空目录
tree	树形结构显示目录，需要安装tree包
pwd	显示当前目录
ln	创建链接文件
more、less	分页显示文本文件内容
head、tail	显示文件头、尾内容
ctrl+alt+F1	命令行全屏模式

系统管理命令

stat	显示指定文件的详细信息，比ls更详细
who	显示在线登陆用户
whoami	显示当前操作用户
hostname	显示主机名
uname	显示系统信息
top	动态显示当前耗费资源最多进程信息
ps	显示瞬间进程状态 ps -aux
du	查看目录大小 du -h /home带有单位显示目录信息
df	查看磁盘大小 df -h 带有单位显示磁盘信息
ifconfig	查看网络情况
ping	测试网络连通

netstat 显示网络状态信息

man 命令不会用了，找男人 如：man ls

clear 清屏

alias 对命令重命名 如：alias showmeit="ps -aux" ，另外解除使用unaliax showmeit

kill 杀死进程，可以先用ps 或 top命令查看进程的id，然后再用kill命令杀死进程。

打包压缩相关命令

gzip：

bzip2：

tar: 打包压缩

-c 归档文件

-x 压缩文件

-z gzip压缩文件

-j bzip2压缩文件

-v 显示压缩或解压缩过程 v(view)

-f 使用档名

例：

tar -cvf /home/abc.tar /home/abc 只打包，不压缩

tar -zcvf /home/abc.tar.gz /home/abc 打包，并用gzip压缩

tar -jcvf /home/abc.tar.bz2 /home/abc 打包，并用bzip2压缩

当然，如果想解压缩，就直接替换上面的命令 tar -cvf / tar -zcvf / tar -jcvf 中的“c”换成“x” 就可以了。

关机/重启机器

shutdown

-r 关机重启

-h 关机不重启

now 立刻关机

halt 关机

reboot　　重启

Linux管道

将一个命令的标准输出作为另一个命令的标准输入。也就是把几个命令组合起来使用，后一个命令除以前一个命令的结果。

例：grep -r "close" /home/* | more　　在home目录下所有文件中查找，包括close的文件，并分页输出。

Linux软件包管理

dpkg (Debian Package)管理工具，软件包名以.deb后缀。这种方法适合系统不能联网的情况下。

比如安装tree命令的安装包，先将tree.deb传到Linux系统中。再使用如下命令安装。

sudo dpkg -i tree_1.5.3-1_i386.deb　　安装软件

sudo dpkg -r tree　　卸载软件

注：将tree.deb传到Linux系统中，有多种方式。VMwareTool，使用挂载方式；使用winSCP工具等；

APT（Advanced Packaging Tool）高级软件工具。这种方法适合系统能够连接互联网的情况。

依然以tree为例

sudo apt-get install tree　　安装tree

sudo apt-get remove tree　　卸载tree

sudo apt-get update　　更新软件

sudo apt-get upgrade

将.**rpm**文件转为.**deb**文件

.rpm为RedHat使用的软件格式。在Ubuntu下不能直接使用，所以需要转换一下。

sudo alien abc.rpm

vim使用

vim三种模式：命令模式、插入模式、编辑模式。使用ESC或i或：来切换模式。

命令模式下：

:q 退出

:q! 强制退出

:wq 保存并退出

:set number 显示行号

:set nonumber 隐藏行号

/apache 在文档中查找apache 按n跳到下一个，shift+n上一个

yyp 复制光标所在行，并粘贴

h(左移一个字符←)、j(下一行↓)、k(上一行↑)、l(右移一个字符→)

用户及用户组管理

/etc/passwd 存储用户账号

/etc/group 存储组账号

/etc/shadow 存储用户账号的密码

/etc/gshadow 存储用户组账号的密码

useradd 用户名

userdel 用户名

adduser 用户名

groupadd 组名

groupdel 组名

passwd root 给root设置密码

su root

su - root

/etc/profile 系统环境变量

bash_profile 用户环境变量

.bashrc 用户环境变量

su user 切换用户，加载配置文件.bashrc

su - user 切换用户，加载配置文件/etc/profile ，加载bash_profile

更改文件的用户及用户组

sudo chown [-R] owner[:group] {File|Directory}

例如：还以jdk-7u21-linux-i586.tar.gz为例。属于用户hadoop，组hadoop

要想切换此文件所属的用户及组。可以使用命令。

sudo chown root:root jdk-7u21-linux-i586.tar.gz

来自 <http://www.cnblogs.com/laov/p/3541414.html>

? 6. nginx命令集合

1. 启动 Nginx　　/usr/local/nginx/sbin/nginx

poechant@ubuntu:sudo ./sbin/nginx

2. 停止 Nginx

poechant@ubuntu:sudo ./sbin/nginx -s stop
poechant@ubuntu:sudo ./sbin/nginx -s quit

-s都是采用向 Nginx 发送信号的方式。
3. Nginx 重载配置

poechant@ubuntu:sudo ./sbin/nginx -s reload

上述是采用向 Nginx 发送信号的方式，或者使用：

poechant@ubuntu:service nginx reload

4. 指定配置文件

poechant@ubuntu:sudo ./sbin/nginx -c /usr/local/nginx/conf/nginx.conf

-c表示configuration，指定配置文件。
5. 查看 Nginx 版本

有两种可以查看 Nginx 的版本信息的参数。第一种如下：

poechant@ubuntu:/usr/local/nginx$ ./sbin/nginx -v
nginx: nginx version: nginx/1.0.0

另一种显示的是详细的版本信息：

poechant@ubuntu:/usr/local/nginx$ ./sbin/nginx -V

nginx: nginx version: nginx/1.0.0
nginx: built by gcc 4.3.3 (Ubuntu 4.3.3-5ubuntu4)
nginx: TLS SNI support enabled
nginx: configure arguments: --with-http_ssl_module --with-openssl=/home/luming/openssl-1.0.0d/

6. 检查配置文件是否正确

poechant@ubuntu:/usr/local/nginx$ ./sbin/nginx -t
nginx: [alert] could not open error log file: open() "/usr/local/nginx/logs/error.log" failed (13: Permission denied)
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
2012/01/09 16:45:09 [emerg] 23898#0: open() "/usr/local/nginx/logs/nginx.pid" failed (13: Permission denied)
nginx: configuration file /usr/local/nginx/conf/nginx.conf test failed

如果出现如上的提示信息，表示没有访问错误日志文件和进程，可以sudo（super user do）一下：

poerchant@ubuntu:/usr/local/nginx$ sudo ./sbin/nginx -t
nginx: the configuration file /usr/local/nginx/conf/nginx.conf syntax is ok
nginx: configuration file /usr/local/nginx/conf/nginx.conf test is successful

如果显示如上，则表示配置文件正确。否则，会有相关提示。
7. 显示帮助信息

来自 <http://www.51testing.com/html/71/410671-842861.html>

7. **linux中 | 的作用**
8. linux查看内存，查看磁盘，查看内核信息
   Top
9.