Linux中find常见用法示例

·find path -option [-print] [-exec -ok command] {} \;

find命令的参数；

pathname: find命令所查找的目录路径。例如用.来表示当前目录，用/来表示系统根目录。
-print： find命令将匹配的文件输出到标准输出。
-exec： find命令对匹配的文件执行该参数所给出的shell命令。相应命令的形式为'command' { } \;，注意{ }和\；之间的空格。
-ok： 和-exec的作用相同，只不过以一种更为安全的模式来执行该参数所给出的shell命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

#-print 将查找到的文件输出到标准输出
#-exec command {} \; ——将查到的文件执行command操作,{} 和 \;之间有空格
#-ok 和-exec相同，只不过在操作前要询用户

例：find . -name .svn | xargs rm -rf

==========

-name filename #查找名为filename的文件
-perm #按执行权限来查找
-user username #按文件属主来查找
-group groupname #按组来查找
-mtime -n +n #按文件更改时间来查找文件，-n指n天以内，+n指n天以前
-atime -n +n #按文件访问时间来查GIN: 0px">

-ctime -n +n #按文件创建时间来查找文件，-n指n天以内，+n指n天以前

-nogroup #查无有效属组的文件，即文件的属组在/etc/groups中不存在
-nouser #查无有效属主的文件，即文件的属主在/etc/passwd中不存
-newer f1 !f2 找文件，-n指n天以内，+n指n天以前
-ctime -n +n #按文件创建时间来查找文件，-n指n天以内，+n指n天以前
-nogroup #查无有效属组的文件，即文件的属组在/etc/groups中不存在
-nouser #查无有效属主的文件，即文件的属主在/etc/passwd中不存
-newer f1 !f2 #查更改时间比f1新但比f2旧的文件
-type b/d/c/p/l/f #查是块设备、目录、字符设备、管道、符号链接、普通文件
-size n[c] #查长度为n块[或n字节]的文件
-depth #使查找在进入子目录前先行查找完本目录
-fstype #查更改时间比f1新但比f2旧的文件
-type b/d/c/p/l/f #查是块设备、目录、字符设备、管道、符号链接、普通文件
-size n[c] #查长度为n块[或n字节]的文件
-depth #使查找在进入子目录前先行查找完本目录
-fstype #查位于某一类型文件系统中的文件，这些文件系统类型通常可 在/etc/fstab中找到
-mount #查文件时不跨越文件系统mount点
-follow #如果遇到符号链接文件，就跟踪链接所指的文件
-cpio %; #查位于某一类型文件系统中的文件，这些文件系统类型通常可 在/etc/fstab中找到
-mount #查文件时不跨越文件系统mount点
-follow #如果遇到符号链接文件，就跟踪链接所指的文件
-cpio #对匹配的文件使用cpio命令，将他们备份到磁带设备中
-prune #忽略某个目录

==========

$find ~ -name "*.txt" -print #在$HOME中查.txt文件并显示
$find . -name "*.txt" -print

```
$find . -name "[A-Z]*" -print #查以大写字母开头的文件
$find /etc -name "host*" -print #查以host开头的文件
$find . -name "[a-z][a-z][0–9][0–9].txt" -print #查以两个小写字母和两个数字开头的txt文件
$find . -perm 755 -print
$find . -perm -007 -exec ls -l {} \; #查所有用户都可读写执行的文件同-perm 777
$find . -type d -print
$find . ! -type d -print
$find . -type l -print

$find . -size +1000000c -print #查长度大于1Mb的文件
$find . -size 100c -print # 查长度为100c的文件
$find . -size +10 -print #查长度超过期作废10块的文件（1块=512字节）

$cd /
$find etc home apps -depth -print | cpio -ivcdC65536 -o /dev/rmt0
$find /etc -name "passwd*" -exec grep "cnscn" {} \; #看是否存在cnscn用户
$find . -name "yao*" | xargs file
$find . -name "yao*" | xargs echo "" > /tmp/core.log
$find . -name "yao*" | xargs chmod o-w

=====================================================
```

find -name april* 在当前目录下查找以**april**开始的文件
find -name april* fprint file 在当前目录下查找以**april**开始的文件，并把结果输出到**file**中
find -name ap* -o -name may* 查找以**ap**或**may**开头的文件
find /mnt -name tom.txt -ftype vfat 在**/mnt**下查找名称为**tom.txt**且文件系统类型为**vfat**的文件
find /mnt -name t.txt ! -ftype vfat 在**/mnt**下查找名称为**tom.txt**且文件系统类型不为**vfat**的文件
find /tmp -name wa* -type l 在**/tmp**下查找名为**wa**开头且类型为符号链接的文件
find /home -mtime -2 在**/home**下查最近两天内改动过的文件
find /home -atime -1 查**1**天之内被存取过的文件
find /home -mmin +60 在**/home**下查**60**分钟前改动过的文件
find /home -amin +30 查最近**30**分钟前被存取过的文件
find /home -newer tmp.txt 在**/home**下查更新时间比**tmp.txt**近的文件或目录
find /home -anewer tmp.txt 在**/home**下查存取时间比**tmp.txt**近的文件或目录
find /home -used -2 列出文件或目录被改动过之后，在**2**日内被存取过的文件或目录
find /home -user cnscn 列出**/home**目录内属于用户**cnscn**的文件或目录
find /home -uid +501 列出**/home**目录内用户的识别码大于**501**的文件或目录
find /home -group cnscn 列出**/home**内组为**cnscn**的文件或目录
find /home -gid 501 列出**/home**内组**id**为**501**的文件或目录
find /home -nouser 列出**/home**内不属于本地用户的文件或目录
find /home -nogroup 列出**/home**内不属于本地组的文件或目录
find /home -name tmp.txt -maxdepth 4 列出**/home**内的**tmp.txt** 查时深度最多为**3**层
find /home -name tmp.txt -mindepth 3 从第**2**层开始查
find /home -empty 查找大小为**0**的文件或空目录
find /home -size +512k 查大于**512k**的文件
find /home -size -512k 查小于**512k**的文件
find /home -links +2 查硬连接数大于**2**的文件或目录
find /home -perm 0700 查权限为**700**的文件或目录
find /tmp -name tmp.txt -exec cat {} \;
find /tmp -name tmp.txt -ok rm {} \;

find / -amin -10 # 查找在系统中最后**10**分钟访问的文件
find / -atime -2 # 查找在系统中最后**48**小时访问的文件
find / -empty # 查找在系统中为空的文件或者文件夹
find / -group cat # 查找在系统中属于 **groupcat**的文件
find / -mmin -5 # 查找在系统中最后**5**分钟里修改过的文件
find / -mtime -1 #查找在系统中最后**24**小时里修改过的文件

find / -nouser #查找在系统中属于作废用户的文件
find / -user fred #查找在系统中属于**FRED**这个用户的文件

查当前目录下的所有普通文件

 # find . -type f -exec ls -l {} \;
-rw-r–r– 1 root root 34928 2003-02-25 ./conf/httpd.conf
-rw-r–r– 1 root root 12959 2003-02-25 ./conf/magic
-rw-r–r– 1 root root 180 2003-02-25 ./conf.d/README
查当前目录下的所有普通文件，并在**- e x e c**选项中使用**ls -l**命令将它们列出

==========

在/ l o g s目录中查找更改时间在5日以前的文件并删除它们：
$ find logs -type f -mtime +5 -exec -ok rm {} \;

==========

查询当天修改过的文件
[root@book class]# find ./ -mtime -1 -type f -exec ls -l {} \;

==========

查询文件并询问是否要显示
[root@book class]# find ./ -mtime -1 -type f -ok ls -l {} \;
< ls … ./classDB.inc.php > ? y
-rw-r–r– 1 cnscn cnscn 13709 1月 12 12:22 ./classDB.inc.php
[root@book class]# find ./ -mtime -1 -type f -ok ls -l {} \;
< ls … ./classDB.inc.php > ? n
[root@book class]#

==========

查询并交给awk去处理
[root@book class]# who | awk ’{print $1"\t"$2}’
cnscn pts/0

==========

awk—grep—sed

[root@book class]# df -k | awk ‘{print $1}’ | grep -v ’none’ | sed s"/\/dev\///g"
文件系统
sda2
sda1
[root@book class]# df -k | awk ‘{print $1}’ | grep -v ’none’
文件系统
/dev/sda2
/dev/sda1

1)在**/tmp**中查找所有的***.h**，并在这些文件中查找**“SYSCALL_VECTOR"**，最后打印出所有包含**"SYSCALL_VECTOR"**的文件名

A) find /tmp -name "*.h" | xargs -n50 grep SYSCALL_VECTOR
B) grep SYSCALL_VECTOR /tmp/*.h | cut -d’:’ -f1| uniq > filename
C) find /tmp -name "*.h" -exec grep "SYSCALL_VECTOR" {} \; -print

2)find / -name filename -exec rm -rf {} \;
find / -name filename -ok rm -rf {} \;

3)比如要查找磁盘中大于**3M**的文件：
find . -size +3000k -exec ls -ld {} ;

4)将find出来的东西拷到另一个地方
find *.c -exec cp ‘{}’ /tmp ‘;’

如果有特殊文件，可以用cpio，也可以用这样的语法：
find dir -name filename -print | cpio -pdv newdir

6)查找**2004-11-30 16:36:37**时更改过的文件
 # A=`find ./ -name "*php"` | ls -l –full-time $A 2>/dev/null | grep "2004-11-30 16:36:37"

Linux-all, Linux | No Comments »

find 实例

四月 18th, 2006

要在/usr/linux中查找所有的*.h，并在这些文件中查找“SYSCALL_VECTOR"，最后打印出所有包含"SYSCALL_VECTOR"的文件名，有以下几种方法实现
find /usr/linux -name "*.h" | xargs -n50 grep SYSCALL_VECTOR
grep SYSCALL_VECTOR /usr/linux/*.h | cut -d':' -f1 | uniq > filename
find /usr/linux -name "*.h" -exec grep "SYSCALL_VECTOR" {} \; -print

我用find / -name filename| rm -rf，不成功，请问为什么不成功?
find / -name filename -exec rm -rf {} \;
find . -name filename |rm -rf试一下{} 表示你找出来的结果。
\; 则相当于“宪法”，没什么说头，就是这么规定的，在 -exec 后面需要一个表示该命令终结的的符号。可以在 man find 中找到答案。
要让rm识别find的结果，如下：
find / -name filename |xargs rm -rf
之所以find . -name filename |rm -rf不通过，是因为rm命令不接受从标准输入传过来的指令
查找含特定字符串的文件
例如查找当前目录下含有"the string you want find…"字符串的文件：
$find . -type f -exec grep “the string you want find…” {} ; -print

从根目录开始查tmpfile，一旦查到马上删除
find / -name "tmpfile" -exec rm {} \;

find 的perm问题
请问一下以下命令什么意思？关键是那个数字前的-，其他都还知道
find -name ".*" -perm -007
我知道
find -name ".*" -perm 755
这个是用来查找权限位为755的隐藏文件
噢，对了还有，我上边的命令都省略了find的pathname参数 find默认是查找当前工作目录的吗?
如果我用 -ok 替代 -exec, 那么还需要加上 {} \; 吗?
这个已经清楚，仍然需要，因为 -ok 只是 -exec 的提示模式，它只是多了一个确认操作的步骤,刚才没有读懂那几句E文的意思 呵呵 不好意思
-007是指查找所有用户都可读、写、执行的文件，要小心呀~~~
解释解释?
find -name ".*" -perm -007 和 find -name ".*" -perm 777 有区别吗?
-007是怎么来得呢?
不过有一个问题
我用 find . -perm -100 会列出当前目录 . , 这是为什么呢?

下面引用由explover在 2002/10/01 06:15am 发表的内容：
-007是指查找所有用户都可读、写、执行的文件，要小心呀~~~
-007是查找含其它用户(不同组,非属主)可读,写,执行的文件.并不一定要同组可读写,-是指最少权限为007.
下面引用由一颗小白菜在 2002/10/01 10:16am 发表的内容：
OK了， 呵呵
不过有一个问题
我用 find . -perm -100 会列出当前目录 . , 这是为什么呢?
这种方法不会准确的找出目录的. -100是指权限至少是属主可运行.
在unix系统下,你可以拥有对目录文件的执行权你才可以进入一个目录.这便是目录文件被列出的原因.
find . -perm -001 -print找到往往是目录文件.
我的意思当然不是使用这种方法来找目录，只不过不明白其中的 -100 意义了

那以此类推，是不是 -010是指权限至少是owner同组可执行的吗？也就是说其实这里的010和-是分开的，-表示一个至少的意思，而且010才是真正用来描述权限位的？
这样子就明白了 谢谢你噢

将find出来的东西拷到另一个地方?
find *.c -exec cp ‘{}’ /tmp ‘;’
如果有特殊文件，可以用cpio，也可以用这样的语法：
find dir -name filename -print | cpio -pdv newdir

找出磁盘中某个大小范围内的文件
比如要查找磁盘中大于3M的文件：
find . -size +3000k -exec ls -ld {} ;

如何用find查找某一天更改的文件?
可以使用这一行命令来实现：
A=`find ~ -print` | ls -l –full-time $A 2>/dev/null | grep "Jun 27" | grep 1998

使用find 命令查找某个时间段的shell怎么写。比如11点到12点的。thanks
创建一个脚本judgetime，内容如下：
ls -l $*|awk ‘{split($8,hour,":");if((hour[1]>23 || hour[1] < 1)&&hour[1]<24)print}’
到要查找的目录下，运行
find ./ -name "*" -exec judgetime {} \;
注意时间格式为２４小时制。
thank you ，如果我要精确到分钟呢
touch -t 04241112 starttemp #精确到12分钟
touch -t 04241220 endtemp #截止到12点20
find [dir] -newer starttemp -a ! -newer endtemp -exec ls -l {} \;
ｎｅｗｅｒ？
那昨天１２：１０文件如何呢?
每天执行的时候，用当天的日期和时间戳替换一下不就行了吗?
我不知道他是不是把所有的11：00~12：00的都找出来，是不是只执行一次还是每天都执行?
这种情况俺猜想是自己的东西放在哪忘了，只记得当时是深夜了。
有道理!
不愧是斑竹!
不光知道怎么解决问题，还知道在什么情况下出现这类问题，佩服佩服!
问题又出现了。创建这个文件的时候。本来应该是时间的一栏现在写上了2002，而不是12：00.
等到12：00过了吧!

删除指定日期的文件
find ./ -name 文件名 -exec rm -f {} \;
例：删除当前30天内没用过的文件,用如下命令：
find / -atime +30 -exec rm -f {} \;
我自己试着写了一小段SHELL,也用ll ,grep, rm 几个命令，用起来还差强人意。
对过滤出来的文件名列表中用了一个FOR语句，再执行rm 。现在我想把这段SHELL 扩展一下让它每天定时运行将n 天前的文件删掉，有没有人能给我一些提示，谢谢!
还有个问题，对于前面那位朋友提到的"find / -atime +30 -exec rm -f {} \;
"方法，我很早就试过几次，不过好像都不太对，参数 -atime n 是查找n天前被访问过的文件，我不明白的是这里的时间参照点是什么，以及这个n天是怎么计算的。
问 题二、对于"ll |cut -f 1" 这个命令我是不是用错了，我只想取出 ll 中列出的文件名，但用cut -f 命令做不到 ，我只好换用 ll |cut -c 59- 这种方式得到我要的文件名，but it’s a pool idear ！ 我也试过用awk ，好像也不对，看看大家可不可以给我一些小小的提醒，TKS SO MUCH
问题三、如何改变 l结点 的日期格式 我现在的系统显示的格式是：
-rw-r—— 1 msahz01 users 2253 2002年2月 2日 poheader.i
我想把这换成
-rw-rw-rw- 1 house users 2193 Apr 19 2001 hkdisp.p
如何才能做到这点?

awk 应该可以
ll | awk '{print $9}'
删除多少天之前的文件
find /yourpath -mtime +31 -exec rm {} \;
find /yourpath -mtime +366 -exec rm {} \;

find中, -ctime, -mtime及其-atime有何区别

请问 -ctime 和 -mtime 有什么关系 ?
如果父目录的 ctime 改变, 那它下面的文件的 ctime 就会自动都改了吗 ?
-ctime 和 -mtime ,-atime 这些信息是存在哪儿呢 ?

我用 -mtime -1 找到了新建或改的文件.
但怎样才能找到一天内 mv 来的文件呢(它们的时间是原有的时间,早于一天) ?

用-newer选项啊。
你可以先touch一个你想要的时间的文件如下:
$ touch -t 08190800 test
$ ls -l test
-rw-r–r– 1 dba other 0 Aug 19 08:00 test
然后
$ find . -newer test -print
.
./.sh_history
$ ls -l .sh_history
-rw——- 1 dba other 154 Aug 20 17:39 .sh_history

用touch可以写出你想要的任何时间的文件，然后用-newer ,! -newer选项即可成功。

1.ctime含inode信息修改的时间.mtime只指文件内容建立或修改的时间.
2 不会.
3.这些信息应该是存在文件系统的超级块里.

我查了书 -ctime 是指 inode 的改变(或称文件的状态改变).
请问 inode 存了哪些信息 ?
做了些小测试,-mtime 改, -ctime 一定也改.
改文件名, -ctime 也会改.
谁能回答 i-node 存了哪些东西 ?

vi /usr/include/sys/inode.h

班主,我不能 access /usr/include/sys/inode.h .
摘书如下:
Directories contain directory entries. Each entry contains a file or subdirectory name and an index node reference number (i-node number). To increase speed and enhance use of disk space, the data in a file is stored at various locations in the computer's memory. The i-node contains the addresses used to locate all the scattered blocks of data associated with a file. The i-node also records other information about the file including time of modification and access, access modes, number of links, file owner, and file type.
可我发现 -atime 改了, -ctime 还没改. why ?
(我先 cat 一个 ASCII 文件,再用 -atime -1 有它用 -ctime -1 居然没有它.)
着岂不跟 inode 信息改变, ctime 就改矛盾吗?

我不同意你贴出来的那段文章,正如我提到的那样,atime,ctime,mtime是放到超级块里,在sco unix下是一种叫stat的结构.(stat_32),不同的系统文件系统可能不同.
sco 下inode的结构如下:

```
typedef struct inode
{
struct inode *i_forw; /* inode hash chain */
struct inode *i_back; /* '' */
struct inode *av_forw; /* freelist chain */
struct inode *av_back; /* '' */
```

```
int *i_fsptr; /* "typeless" pointer to fs dependent */
ino32_t i_number; /* i number, 1-to-1 with dev address */
ushort i_ftype; /* file type = IFDIR, IFREG, etc. */
short i_fstyp; /* File system type */
off_t i_size; /* size of file */
ushort i_uid; /* owner */
ushort i_gid; /* group of owner */
ushort i_flag;
ushort i_want; /* i_flag extension to avoid MP races */
ushort i_count; /* reference count */
short i_nlink; /* directory entries */
dev_t i_rdev; /* Raw device number */
#define i_namtype i_rdev /* i_ftype==IFNAM subtype */
dev_t i_dev; /* device where inode resides */
struct mount *i_mton;/* ptr to mount table entry that */
/* this directory is mounted on */
struct region *i_rp; /* ptr to shared region if any */
struct stdata *i_sp; /* ptr to associated stream */
struct iisem *isem; /* ptr to XENIX semaphores */
struct iisd *isd; /* ptr to XENIX shared data */
} i_un;
#define i_mnton i_un.i_mton /* i_ftype==IFDIR IMOUNT */
#define i_rptr i_un.i_rp /* i_ftype==IFREG || i_ftype==IFBLK */
#define i_sptr i_un.i_sp /* i_ftype==IFCHR || i_ftype==IFIFO */
#define i_sem i_un.isem /* i_ftype==IFNAM && i_namtype==IFSEM */
#define i_sd i_un.isd /* i_ftype==IFNAM && i_namtype==IFSHD */

struct fstypsw *i_fstypp; /* ptr to file system switch FSPTR */
long *i_filocks; /* pointer to filock (structure) list */
unsigned long i_mappages; /* number of pages currently cached */
unsigned long i_vcode; /* read-ahead block save (NFS) */
short i_wcnt; /* write open count or ITEXT count */
struct lockb i_cilock; /* tas to synchronize i_flag changes */
ushort i_rdlocks; /* count of non-exclusive lockers */
} inode_t;
```

所以,访问一个文件不能改变inode信息.
使用chown, chgrp, chmod命令可以很好的比较mtime和ctime
chown改变一个文件的属主,用ctime可以找到,用mtime便找不到.
试试看.

多谢斑竹! 我是在 Solaris 上面试的.我是对 -ctime 不明白.
试的结果如下:
修改文件,-mtime 改了, -ctime 也会改.
访问文件,-atime 改了, -ctime 没变.
chown, chgrp, chmod,mv, 都会使 -ctime 改变,但不影响 -atime 和 -mtime.
touch 可以改 -mtime and/or -atime,但 touch -a 只改访问时间时,-ctime也改了.
touch -m 改修改时间时,-ctime当然也改了.
好象还有别的很多东西可以令 -ctime 改变, 搞不清楚.
有什么方法可以显示 -mtime,atime,ctime 吗?
可以用 -ctime 来实现对目录的增量文件进行备份或 transfer 吗 ?
多谢!

没有什么工具显示,(可能是俺不知道)
把下面程序里的st_mtime换成st_ctime,或st_atime便可以得到你要的了.

```
#include
int
main (int argc, char **argv)
{
struct stat buf;
char date[80];
char fname[80];
```

```
printf("Enter filename (with full path) to check mtime : ");
scanf("%s",fname);
stat(fname, &buf);
printf ("mtime (in sec) of %s = %ld\n", fname, buf.st_mtime);
strcpy(date, ctime((time_t *)&(buf.st_mtime)));
printf ("mtime (in date) of %s = %s\n", fname, date);
}
```

至于文件备份,有什么不可以的么?

mtime ls -l 最近修改文件内容的时间
atime ls -lu 最近访问文件的时间
ctime ls -li 最近文件有所改变的状态 ,如文件修改,属性\属主 改变 ,节点 ,链接变化等 ,应该是不拘泥只是时间前后的改变

俺看了ls的帮助,以为只是按ctime或atime排序,显示的时间还是mtime.

仔细比较了一下,ayhan说的是对的.谢谢ayhan.

多谢 ahyan 提示 ! 我在 Solaris 上试过如下:
mtime 用 ls -l 看到
atime 用 ls -lu 看到
ctime 用 ls -lc 看到. (ls -li 只有 inode number)
摘书如下:
-c Uses time of last modification of the i-node (file created, mode changed, and so forth) for sorting (-t) or printing (-l or -n).
-u Uses time of last access instead of last modification for sorting (with the -t option) or printing (with the -l option).
-i For each file, prints the i-node number in the first column of the report.