# Summer 2013 Handout 10

**Command line arguments for find: see `find`(1)**

The first argument is always the name of a directory. **`find`** will search this directory, its children, grandchildren, etc. **`-print`** will output the full pathname of each thing that **`find`** finds.

```
1$ find ~ -name index.html -print        Find anything named index.html. -print is optional.
2$ find ~ -name index.html -ls           minus lowercase LS: just like ls -li
3$ find ~ -name '*.html' -print          Need quotes; can also use ? [ ] (Handout 8, p. 16)
4$ find ~ -type f -print                 find every file: f for file, d for directory
```

**Search the entire tree of directories**

You can lower the **`find`**'s priority with **`nice`**.

```
1$ nice cal 12 2013
```

```
2$ nice sleep 10 & ps -l            minus lowercase L: NI value higher than default of 20
 F S    UID   PID  PPID   C PRI NI     ADDR     SZ    WCHAN TTY          TIME CMD
 0 S  50766 21836 17654   0  98 34        ?    158        ? pts/1        0:00 sleep
 0 O  50766 21837 17654   0  98 20        ?    184          pts/1        0:00 ps
 0 S  50766 17654 17652   0  98 20        ?    257        ? pts/1        0:01 ksh
```

**`find`** can search only those directories for which you have both **`r`** and **`x`** permission. Use **`2> /dev/null`** to discard the error message that **`find`** issues when it is rebuffed by a directory for which you do not have both permissions. For **`2>`**, see Handout 5, p. 8; for **`/dev/null`**, see Handout 2, pp. 20−21. Run the **`find`** in the background with an ampersand (Handout 6, p. 6).

```
3$ nice find / -name index.html -print > find.out 2> /dev/null &
4$                                  The prompt reappears immediately.
```

**and, or, not**

Two consecutive conditions (such as **`-type f`** and **`-name index.html`**) are assumed to have an implicit **`-a`** (for ''and'') between them. For ''or'', you must write an explicit **`-o`** between them.

```
1$ find ~ -type f -a -name index.html -print     files named index.html
2$ find ~ -type f    -name index.html -print     -a is optional, -o is mandatory
3$ find ~ -type d    -name '*bin*' -print        directories whose name contains bin
```

Exclamation point means ''not''. Surprisingly, **`!`** is not a special character in the Korn and Bourne shells, and is not a special character when followed by a blank in the C shell.

```
4$ find ~   -user abc1234 -print          anything owned by abc1234
5$ find ~ ! -user abc1234 -print          anything not owned by abc1234

6$ find ~   -perm  644 -print             anything with exactly rw-r--r--
7$ find ~   -perm -644 -print             anything with at least rw-r--r--
8$ find ~ ! -perm -644 -print             anything with less than rw-r--r--
```

```
9$ find / -perm -4100 -ls 2> /dev/null      anything with set-uid and left x bits on
116706  109 -r-sr-xr-x   1 nagios   nagios     192144 Oct 10  2012 /local/nagios/libexe
116694  109 -r-sr-xr-x   1 nagios   nagios     196100 Oct 10  2012 /local/nagios/libexe
116730   13 -r-sr-xr-x   1 nagios   nagios      25752 Oct 10  2012 /local/nagios/libexe

10$ find / -perm -4000 ! -perm -100 -ls 2> /dev/null   anything with set-uid on and left x off
11485    1 -r-Sr-Sr--   1 mm64     users           0 Jan 27  1998 /home1/m/mm64/public_
 3548    2 drwSrw-rwT   2 mm64     users           2 Jan 31  2001 /home1/m/mm64/public_
 3548    2 drwSrw-rwT   2 mm64     users           2 Jan 31  2001 /home1/m/mm64/public_
```

Every file in a website must have at least the permissions **r--r--r--** in order to be visible on the web. Every directory must have at least **r-xr-xr-x**. Let's find the files and directories that do not have the required permissions.

```
11$ find ~/public_html -type f ! -perm -444 -print
12$ find ~/public_html -type d ! -perm -555 -print
13$ find ~/public_html -type f ! -perm -444 -o -type d ! -perm -555 -print
```

```
#!/bin/ksh
#Find everything whose permissions are heavier on the right.
#See Handout 6, p. 11.
#-perm -040 means that the middle r is on.
#! -perm -400 means that the left r is off.
#-perm -020 means that the middle w is on.  Et cetera.

find / \
    -perm -040 -a ! -perm -400 -o \
    -perm -020 -a ! -perm -200 -o \
    -perm -010 -a ! -perm -100 -o \
    -perm -004 -a ! -perm -040 -o \
    -perm -002 -a ! -perm -020 -o \
    -perm -001 -a ! -perm -010 \
    -ls 2> /dev/null
```

```
313    1 -rw-rw-rwx   1 up244    users         270 Aug  1  2012 /home1/u/up244/public_html/homew
  3    2 drwx-----x   5 el83     users          27 Sep  5  2008 /home1/e/el83
 83    2 drwxr--r-x   2 eag235   users          13 Dec 11  2003 /home1/e/eag235/public_html/Comp
 83    2 drwxr--r-x   2 eag235   users          13 Dec 11  2003 /home1/e/eag235/public_html/Comp
  3    2 drwx-----x   9 zl226    users          29 May  5  2003 /home1/z/zl226
```

**Group the arguments with parentheses**

It is dangerous for a script to have its second or third **w** bit on when its set-uid bit is on: **rwsrwxrwx**. **!** has higher precedence than **-a**, which in turn has higher precedence than **-o**. We therefore need parentheses to group together the demands that the second or third **w** bits be on. Without the parentheses, this script would find any file whose set-uid and second **w** bit is on, or anything at all whose third **w** bit is on. Since parentheses are special characters for the shell, each parenthesis must be in **'**single quotes**'**,

```
#!/bin/ksh
#Find any file whose set-uid is on and whose 2nd or 3rd w bit is on.

find / \
    -type f \
    -perm -4000 \
    '(' -perm -020 -o -perm -002 ')' \
    -ls 2> /dev/null
```

**http://i5.nyu.edu/~mm64/INFO1-CE9545/src/vulnerable**

```
#!/bin/ksh
#Find any file or directory whose 2nd or 3rd w bit is on.
#These are the files that can be overwritten,
#and the directories from which files can be stolen.

for loginname in `~mm64/bin/roster 45 | grep -v '^mm64$'`
do
    find ~$loginname \
        '(' -type f -o -type d ')' \
        '(' -perm -020 -o -perm -002 ')' \
        -ls 2> /dev/null
done

exit 0
```

```
1$ vulnerable | head -5
    3    3 drwxrwxrwx   4 yb610    users    35 Aug  5  2012 /home1/y/yb610
   54    1 -rwxrwxrwx   1 up244    users   197 Jun 27  2012 /home1/y/yb610/Homework-3.8
   13    1 -rwxrwxrwx   1 yb610    users   133 May 29  2012 /home1/y/yb610/yb610
   56    4 -rwxrwxrwx   1 up244    users  2612 Jun 27  2012 /home1/y/yb610/homework4.1
   89    1 -rwxrwxrwx   1 yb610    users   297 Jul 17  2012 /home1/y/yb610/bin/GIDnumber
```

```
2$ find ~ -type f -a -size  1000c -print      files whose size is exactly 1000 bytes ("characters")
3$ find ~ -type f -a -size +1000c -print      files whose size is greater than 1000 bytes
4$ find ~ -type f -a -size -1000c -print      files whose size is less than 1000 bytes
```

What's the simplest way to **find** all files whose size is greater than or equal to 1000 bytes?

```
5$ find ~ -type f -a -atime  10 -print        files last accessed 10 days ago: ls -lu
6$ find ~ -type f -a -atime +10 -print        files last accessed more than 10 days ago
7$ find ~ -type f -a -atime -10 -print        files last accessed less than 10 days ago

8$ find ~ -type f -a -mtime +10 -print        files last modified more than 10 days ago: ls -lt
9$ find ~ -type f -a -ctime +10 -print        files last changed more than 10 days ago: ls -lc
```

"Accessed" means that the file was input into a program. "Modified" means that a program's output was deposited into the file, i.e., that the contents of the file were changed. "Changed" means that the contents of the file were changed, or that some of the facts displayed by **ls -l** were changed: the nine permission bits (changed by **chmod**), the owner of the file (changed by **chown**), etc.

```
10$ date > junk1
11$ date > junk2
12$ date > junk3                              junk3 was most recently modified.
```

```
13$ chmod 400 junk2                    junk2 was most recently changed.

14$ cat junk3
15$ cat junk1                          junk1 was most recently accessed.
```

The **-r** option of **ls** ("reverse", Handout 1, p. 10) can be combined with any of the following.

```
16$ ls -l junk[123]                    alphabetical order, showing modification times
-rw-------   1 abc1234  users          29 May 28 15:50 junk1
-r--------   1 abc1234  users          29 May 28 15:51 junk2
-rw-------   1 abc1234  users          29 May 28 15:52 junk3

17$ ls -ltu junk[123]                      accessed order, showing access times
-rw-------   1 abc1234  users          29 May 28 15:55 junk1
-rw-------   1 abc1234  users          29 May 28 15:54 junk3
-r--------   1 abc1234  users          29 May 28 15:51 junk2

18$ ls -lt junk[123]                   modified order, showing modification times
-rw-------   1 abc1234  users          29 May 28 15:52 junk3
-r--------   1 abc1234  users          29 May 28 15:51 junk2
-rw-------   1 abc1234  users          29 May 28 15:50 junk1

19$ ls -ltc junk[123]                      changed order, showing changed times
-r--------   1 abc1234  users          29 May 28 15:53 junk2
-rw-------   1 abc1234  users          29 May 28 15:52 junk3
-rw-------   1 abc1234  users          29 May 28 15:50 junk1
```

**Use the output of find as the command line arguments of another command**

The **post** shellscript in Handout 4, p. 23 can take one or more filenames as command line arguments.

```
1$ rm    `find / -type f -a -user abc1234 -print 2> /dev/null`     Remove the files belonging to abc1234.
2$ rmdir `find / -type d -a -user abc1234 -print 2> /dev/null`     Remove the directories belonging to abc1234.
3$ post `find /some/directory -type f -a -name '*.html' -print 2> /dev/null`
```

The **post** shellscript in Handout 4, pp. 21−22 can take only one filename as a command line argument.

```
#!/bin/ksh
#Post the files whose names end with .html

for filename in `find /some/directory -type f -a -name '*.html' -print \
    2> /dev/null`
do
    post $filename
done
```

The following command does the same thing as the above loop. The command in the **loop** is written between the **-exec** argument and the semicolon argument.

```
4$ find /some/directory -type f -a -name '*.html' -exec post {} ';' \
    2> /dev/null
```

**Find all hard links to a file**

A file can have many names, but it can have only one inode number. A file can be renamed, but its inode number never changes. See Handout 2, p. 6. To find all the hard links to a file, search only within one filesystem.

```
1$ cd /etc/init.d
2$ pwd

3$ ls -li apache

4$ find /etc -inum 0 -ls 2> /dev/null
```

See **init**(1M) for the eight runlevels in our version of Unix.

**Seven symbolic links to the same file**

**s1** and **s2** contain a full pathname; the other five symbolic links contain a relative pathname. It's hard to recognize that all seven are links to the same file:

```
1$ cd
2$ pwd
/home1/a/abc1234

3$ date > junk
4$ ln -s /home1/a/abc1234/junk s1
5$ ln -s /home1/a//abc1234/junk s2
6$ ln -s junk s3
7$ ln -s ./junk s4
8$ ln -s ./././junk s5
9$ ln -s ../abc1234/junk s6
10$ ln -s ../../a/abc1234/junk s7

11$ ls -l /home1/a/abc1234/junk /home1/a/abc1234/s[1-7]
-rw-------   1 abc1234     users         29 May 28 15:55 /home1/a/abc1234/junk
lrwxrwxrwx   1 abc1234     users         29 May 28 15:55 /home1/a/abc1234/s1 -> /home1/a/abc1234/ju
lrwxrwxrwx   1 abc1234     users         30 May 28 15:55 /home1/a/abc1234/s2 -> /home1/a//abc1234/j
lrwxrwxrwx   1 abc1234     users          4 May 28 15:55 /home1/a/abc1234/s3 -> junk
lrwxrwxrwx   1 abc1234     users          6 May 28 15:55 /home1/a/abc1234/s4 -> ./junk
lrwxrwxrwx   1 abc1234     users          8 May 28 15:55 /home1/a/abc1234/s5 -> ./././junk
lrwxrwxrwx   1 abc1234     users         15 May 28 15:55 /home1/a/abc1234/s6 -> ../abc1234/junk
lrwxrwxrwx   1 abc1234     users         20 May 28 15:55 /home1/a/abc1234/s7 -> ../../a/abc1234/ju
```

**Find all symbolic links to a file**

Let's find all symbolic links to the Bourne Shell **/sbin/sh**. The **-a** option ("autosplit") tells Perl to store the fields of each line of input into **$F[0]**, **$F[1]**, **$F[2]**, etc. They're just like **awk**'s **$1**, **$2**, **$3**, etc. The **-n** option tells Perl not to print each line of input by default. The **-e** option tells Perl that the following argument is the list of things for Perl to do. Combine the three options into **-ane**.

 

**http://i5.nyu.edu/~mm64/INFO1-CE9545/src/finder**

```
#!/bin/ksh
#Find every symbolic link leading to $1.

if [[ $# -ne 1 ]]
then
    echo $0: arg is file to search for 1>&2
    exit 1
fi

find / -type l -ls 2> /dev/null |      #lowercase L, minus lowercase LS
perl -ane '
    $target = $F[$#F];             #last field

    if ($target !~ m:^/:) {        #if $target is a relative pathname
        $name = $F[$#F - 2]; #3rd from last field is name of symb link
        $name =~ s:[^/]+$::;       #remove rightmost word
        $target = $name . $target; #now target is full pathname
    }

    $target =~ s:/(\.?/)*:/:g;
    while ($target =~ s:/[^/]+/*\.\.::g) {
    }

    if ($target eq "'$1'") {
        print $_;           #$_ is the current line of input.
    }
'

exit 0
```

```
1$ $S45/finder /sbin/sh | more
```

**Search only one directory with "minus prune"**

To search only one directory, prune every other directory.

```
1$ dirname=name of directory to search
2$ echo $dirname

3$ find $dirname ! -name $dirname -prune \
    -atime +10 -print
```

We need the double quotes to prevent shell wildcards from expanding. If we said **"$*"** instead of **"$@"**, and gave the shellscript **-name a*** as its last two arguments, the **-name a*** would have been given to **find** as one big argument with an embedded space.

**http://i5.nyu.edu/~mm64/INFO1-CE9545/src/find1**

```
#!/bin/ksh
set -x
#Search one directory.
#First argument is name of directory.
#Remaining arguments describe what you're looking for.

if [[ $# -lt 2 ]]
then
        echo $0: needs at least 2 arguments. 1>&2
fi

dirname=$1
shift
find $dirname ! -name $dirname -prune '(' "$@" ')'
```

**▼ Homework 10.1: find things**

In each case, the last command line argument of **find** should be **-ls**. Shown below is (a reasonable amount of) the output of **find**

(1) C programmers: **find** every file named **stdio.h**.

```
111430    2 -rw-r--r--   1 root     sys          1300 Aug 14  2010 /opt/solstudio12.2/prod/include/
104218    3 -rw-r--r--   1 root     root         3473 Jul  6  2011 /opt/gcc453/lib/gcc/sparc-sun-so
104882    2 -rw-r--r--   1 root     root         1210 Jul  6  2011 /opt/gcc453/include/c++/4.5.3/tr
58555    3 -r--r--r--   1 root     bin          3473 Jul 11  2012 /usr/gcc/4.5/lib/gcc/sparc-sun-so
58366    2 -r--r--r--   1 root     bin          1210 Jul 11  2012 /usr/gcc/4.5/include/c++/4.5.2/tr
86970    7 -rw-r--r--   1 root     bin         12667 Aug 21  2012 /usr/include/stdio.h
11883    2 -rw-r--r--   1 root     bin          1696 Jul  5  2012 /usr/include/ast/stdio.h
```

(2) C++ programmers: **find** every file named **iostream** or **iostream.h**.

```
110889    1 -rw-r--r--   1 root     sys           816 Aug 14  2010 /opt/solstudio12.2/prod/include/
111608    1 -rw-r--r--   1 root     sys           569 Aug 14  2010 /opt/solstudio12.2/prod/include/
111636    1 -rw-r--r--   1 root     sys           272 Aug 14  2010 /opt/solstudio12.2/prod/include/
111371    2 -rw-r--r--   1 root     sys          2701 Aug 14  2010 /opt/solstudio12.2/prod/include/
111681    1 lrwxrwxrwx   1 root     root           17 Oct  7  2012 /opt/solstudio12.2/prod/include/
111665   19 -rw-r--r--   1 root     sys          43948 Aug 14  2010 /opt/solstudio12.2/prod/include/
91324    2 -rw-r--r--   1 bin      bin          1891 Feb 15  2012 /usr/local/include/c++/3.4.6/back
```

(3) **find** every directory named **font**.

```
  332    2 drwxr-xr-x   3 root     bin             4 Jul  5  2012 /usr/lib/font
 1176    2 drwxr-xr-x   6 root     bin             6 Jul  5  2012 /usr/share/groff/1.19.2/font
65269    2 drwxr-xr-x   2 root     sys             3 Jul 19  2012 /lib/svc/manifest/application/for
```

(4) Find every file whose set-uid bit (Handout 9, pp. 10−11) is turned on (use **-perm -4000**) and that is owned by **root**.

```
78936 1976 -rwsr-xr-x   1 root     bin       3693832 Sep 27  2011 /opt/tivoli/tsm/client/ba/bin/dsm
78764 1976 -rwsr-xr-x   1 root     bin       3693832 Sep 27  2011 /opt/tivoli/tsm/client/api/bin64/
88808   20 -r-sr-xr-x   1 root     bin         32164 Aug 27  2012 /usr/xpg4/bin/crontab
88802   34 -rwsr-xr-x   1 root     sys         54756 Aug 27  2012 /usr/xpg4/bin/at
65174    7 -rwsr-xr-x   1 root     adm         12332 Jul 19  2012 /usr/lib/acct/accton
```

(5) **find** every directory named **man** except in **/home1** and its subdirectories. Use **-prune**.

```
109586    2 drwxr-xr-x   8 root     sys             8 Oct  7  2012 /opt/solstudio12.2/p
109559    2 drwxr-xr-x   9 root     sys             9 Oct  7  2012 /opt/solstudio12.2/m
119285    2 drwxr-xr-x   3 root     bin             3 Apr 13 07:26 /opt/csw/share/man
104067    2 drwxr-xr-x   5 root     root            5 Jul  6  2011 /opt/gcc453/share/ma
 1120    2 drwxr-xr-x   3 root     bin             3 Jul 31  2012 /usr/sfw/share/man
```

(6) How many files have names that end with **`.html`**?

```
17303
```

How about **`.htm`**?

```
1618
```

(7) How many directories are you allowed to visit on i5.nyu.edu?

```
23598
```

How many directories are there at each level?

```
1     1
2     21
3     344
4     3453
5     2564
6     2213
7     2154
8     6962
9     1887
10    1606
11    1189
12    506
13    240
14    229
15    151
16    75
17    22
18    6
```

How many files are there on i5.nyu.edu?

```
175350
```

(8) What is the inode number (Handout 2, p. 6) of the file **`/etc/init.d/mysql.server`**? Find every hard link to that file.

▲

□