

# A Quick Start for lsof

Vic Abell

## Introduction

Agreed, the lsof man page is dense and lsof has a plethora of options. There are examples, but the manual page format buries them at the end. How does one get started with lsof?

This file is an attempt to answer that question. It plunges immediately into examples of lsof use to solve problems that involve looking at the open files of Unix processes.

## Finding Uses of a Specific Open File

Often you're interested in knowing who is using a specific file. You know the path to it and you want lsof to tell you the processes that have open references to it.

Simple -- execute lsof and give it the path name of the file of interest -- e.g.,

```
$ lsof /etc/passwd
```

Caveat: this only works if lsof has permission to get the status (via stat(2)) of the file at the named path. Unless the lsof process has enough authority -- e.g., it is being run with a real User ID (UID) of root -- this AIX example won't work:

Further caveat: this use of lsof will fail if the stat(2) kernel syscall returns different file parameters -- particularly device and inode numbers -- than lsof finds in kernel node structures. This condition is rare and is usually documented in the 00FAQ file of this lsof distribution.

```
$ lsof /etc/security/passwd
lsof: status error on /etc/security/passwd: Permission denied
```

## Finding Open Files Filling a File System

Oh! Oh! /tmp is filling and ls doesn't show that any large files are being created. Can lsof help?

Maybe. If there's a process that is writing to a file that has been unlinked, lsof may be able to discover the process for you. You ask it to list all open files on the file system where /tmp is located.

Sometimes /tmp is a file system by itself. In that case,

```
$ lsof /tmp
```

is the appropriate command. If, however, /tmp is part of another file system, typically /, then you may have to ask lsof to list all files open on the containing file system and locate the offending file and its process by inspection -- e.g.,

```
$ lsof / | more or $ lsof / | grep ...
```

Caveat: there must be a file open to a for the lsof search to succeed. Sometimes the kernel may cause a file reference to persist, even where there's no file open to a process. (Can you say kernel bug? Maybe.) In any event, lsof won't be able to help in this case.

## Finding Processes Blocking Umount

When you need to unmount a file system with the umount command, you may find the operation blocked by a process that has a file open on the file systems. lsof may be able to help you find the process. In response to:

# A Quick Start for lsof

Vic Abell

```
$ lsof <file_system_name>
```

Lsof will display all open files on the named file system. It will also set its exit code zero when it finds some open files and non-zero when it doesn't, making this type of lsof call useful in shell scripts. (See section 16.)

Consult the output of the df command for file system names.

See the caveat in the preceding section about file references that persist in the kernel without open file traces. That situation may hamper lsof's ability to help with umount, too.

## Finding Listening Sockets

Sooner or later you may wonder if someone has installed a network server that you don't know about. Lsof can list for you all the network socket files open on your machine with:

```
$ lsof -i
```

The -i option without further qualification lists all open Internet socket files. You can add network names or addresses, protocol names, and service names or port numbers to the -i option to refine the search. (See the next section.)

## Finding a Particular Network Connection

When you know the source or destination of a network connection whose open files and process you'd like to identify, the -i option may help.

If, for example, you want to know what process has a connection open to or from the Internet host named aaa.bbb.ccc, you can ask lsof to search for it with:

```
$ lsof -i@aaa.bbb.ccc
```

If you're further interested in a particular protocol -- TCP or UDP -- and a specific port number or service name, you can add those discriminators to the -i information:

```
$ lsof -iTCP@aaa.bbb.ccc:ftp-data
```

## Identifying a Netstat Connection

How do I identify the process that has a network connection described in netstat output? For example, if netstat says:

```
Proto Recv-Q Send-Q Local Address      Foreign Address   (state)
tcp      0    0 vic.1023          ipscgate.login    ESTABLISHED
```

What process is connected to service name ``login" on ipscgate?

Use lsof's -i option:

```
$lsof -iTCP@ipscgate:login
COMMAND  PID  USER  FD  TYPE   DEVICE  SIZE/OFF  INODE  NAME
rlogin   25023  abe   3u  inet  0x10144168  0t184  TCP
vic.cc.purdue.edu:1023->ipscgate.cc.purdue.edu:login
...
```

## A Quick Start for lsof

Vic Abell

There's another way. Notice the 0x10144168 in the DEVICE column of the lsof output? That's the protocol control block (PCB) address. Many netstat applications will display it when given the -A option:

```
$ netstat -A
PCB  Proto Recv-Q Send-Q Local Address  Foreign Address (state)
10144168 tcp    0    0 vic.1023      ipscgate.login ESTABLISHED
...
```

Using the PCB address, lsof, and grep, you can find the process this way, too:

```
$ lsof -i | grep 10144168
rlogin 25023 abe 3u inet 0x10144168 0t184 TCP
vic.cc.purdue.edu:1023->ipscgate.cc.purdue.edu:login
...
```

### Finding Files Open to a Named Command

When you want to look at the files open to a particular command, you can look up the PID of the process running the command and use lsof's -p option to specify it.

```
$ lsof -p <PID>
```

However, there's a quicker way, using lsof's -c option, provided you don't mind seeing output for every process running the named command.

```
$ lsof -c <first_characters_of_command_name_that_interest_you>
```

The lsof -c option is useful when you want to see how many instances of a given command are executing and what their open files are. One useful example is for the sendmail command.

```
$ lsof -c sendmail
```

### Deciphering the Remote Login Trail

If the network connection you're interested in tracing has been initiated externally and is connected to an rlogind or telnetd process, asking lsof to identify that process might not give a wholly satisfying answer. The report may be that the connection exists, but to a process owned by root.

### The Fundamentals

How do you get from there to the login name really using the connection? You have to know a little about how real and pseudo ttys are paired in your system, and then use several lsof probes to identify the login.

This example comes from a Solaris 2.4 system, named klaatu.cc. I've logged on to it via rlogin from vic.cc. The first lsof probe,

```
$ lsof -i@vic.cc
```

yields (among other things):

```
COMMAND  PID  USER  FD  TYPE  DEVICE  SIZE/OFF  INODE  NAME
in.rlogin 7362  root  0u  inet  0xfc0193b0  0t242  TCP
klaatu.cc.purdue.edu:login->vic.cc.purdue.edu:1023
```

# A Quick Start for lsof

Vic Abell

...

This confirms that a connection exists. A second lsof probe shows:

```
$ lsof -p7362
COMMAND  PID    USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
...
in.rlogin 7362   root  0u  inet  0xfc0193b0  0t242  TCP
klaatu.cc.purdue.edu:login->vic.cc.purdue.edu:1023
...
in.rlogin 7362   root  3u  VCHR  23, 0    0t66 52928
/devices/pseudo/clone@0:ptmx->pckt->ptm
```

7362 is the Process ID (PID) of the in.rlogin process, discovered in the first lsof probe. (I've abbreviated the output to simplify the example.) Now comes a need to understand Solaris pseudo-ttys. The key indicator is in the DEVICE column for FD 3, the major/minor device number of 23,0. This translates to /dev/pts/0, so a third lsof probe,

```
$ lsof /dev/pts/0
COMMAND  PID    USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
ksh      7364   abe   0u  VCHR  24, 0    0t2410 53410
/dev/pts/../../../../devices/pseudo/pts@0:0
```

shows in part that login abe has a ksh process on /dev/pts/0. (The NAME that lsof shows is not /dev/pts/0 but the full expansion of the symbolic link that lsof finds at /dev/pts/0.)

Here's a second example, done on an HP-UX 9.01 host named ghg.ecn. Again, I've logged on to it from vic.cc, so I start with:

```
$ lsof -i@vic.cc
COMMAND  PID    USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
rlogind  10214  root  0u  inet  0x041d5f00  0t1536  TCP
ghg.ecn.purdue.edu:login->vic.cc.purdue.edu:1023
...
```

Then,

```
$ lsof -p10214
COMMAND  PID    USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
...
rlogind  10214  root  0u  inet  0x041d5f00  0t2005  TCP
ghg.ecn.purdue.edu:login->vic.cc.purdue.edu:1023
...
rlogind  10214  root  3u  VCHR  16,0x000030  0t2037 24642 /dev/ptym/ptys0
```

Here the key is the NAME /dev/ptym/ptys0. In HP-UX 9.01 tty and pseudo tty devices are paired with the names like /dev/ptym/ptys0 and /dev/pty/ttys0, so the following lsof probe is the final step.

```
$ lsof /dev/pty/ttys0
COMMAND  PID    USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
ksh      10215  abe   0u  VCHR  17,0x000030  0t3399 22607 /dev/pty/ttys0
...
```

# A Quick Start for lsof

Vic Abell

Here's a third example for an AIX 4.1.4 system. I've used telnet to connect to it from vic.cc.purdue.edu. I start with:

```
$ lsof -i@vic.cc.purdue.edu
COMMAND  PID  USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
...
telnetd  15616  root  0u  inet  0x05a93400  0t5156  TCP
cloud.cc.purdue.edu:telnet->vic.cc.purdue.edu:3369
```

Then I look at the telnetd process:

```
$ lsof -p15616
COMMAND  PID  USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
...
telnetd  15616  root  0u  inet  0x05a93400  0t5641  TCP
cloud.cc.purdue.edu:telnet->vic.cc.purdue.edu:3369
...
telnetd  15616  root  3u  VCHR  25,  0  0t5493  103 /dev/ptc/0
```

Here the key is /dev/ptc/0. In AIX it's paired with /dev/pts/0. The last probe for that shows:

```
$ lsof /dev/pts/0
COMMAND  PID  USER  FD  TYPE   DEVICE  SIZE/OFF  INODE NAME
...
ksh      16642  abe   0u  VCHR  26,  0  0t6461  360 /dev/pts/0
```

## The idrlogin.perl Scripts

There's another, perhaps easier way, to go about the job of tracing a network connection. The lsof distribution contains two Perl scripts, idrlogin.perl (Perl 4) and idrlogin.perl5 (Perl 5), that use lsof field output to display values for shells that are parented by rlogind or telnetd, or connected directly to TCP sockets.

The two Perl scripts use the lsof -R option; it causes the paRent process ID (PPID) to be listed in the lsof output. The scripts identify all shell processes -- e.g., ones whose command names end in `sh' -- and determine if: 1) the ultimate ancestor process before a PID greater than 2 (e.g., init's PID is 1) is rlogind or telnetd; or 2) the shell process has open TCP socket files.

Here's an example of output from idrlogin.perl on a Solaris 2.4 system:

```
centurion: 1 = cd src/lsof4/scripts
centurion: 2 = ./idrlogin.perl
Login Shell  PID Via      PID TTY    From
oboyle ksh      12640 in.telnetd 12638 pts/5  opal.cc.purdue.edu
icdtest ksh      15158 in.rlogind 15155 pts/6  localhost
sh      csh      18207 in.rlogind 18205 pts/1  babylon5.cc.purdue.edu
root    csh      18242 in.rlogind 18205 pts/1  babylon5.cc.purdue.edu
trouble ksh      19208 in.rlogind 18205 pts/1  babylon5.cc.purdue.edu
abe     ksh      21334 in.rlogind 21332 pts/2  vic.cc.purdue.edu
```

The scripts assume that its parent directory contains an executable lsof. If you decide to use one of the scripts, you may want to customize it for your local lsof and perl paths. Note that processes executing as remote shells are also identified.

# A Quick Start for lsof

Vic Abell

Here's another example from a UnixWare 2.1.1 system, where the rlogind action is handled in the kernel -- not by a separate rlogind process -- and the shell's standard I/O descriptors are connected directly to a TCP socket.

```
cyclone: 1 = cd src/lsof4/scripts
cyclone: 2 = ./idrlogin.perl
Login Shell PID Via PID TTY From
abe ksh 3282 (direct) (unknown) vic.cc.purdue.edu
```

In this example the "Via" column says "(direct)" because there is no rlogind process. The PID column for "Via" is empty. The TTY column says "(unknown)" because the shell process has no open tty device file for lsof to report.

## Watching an Ftp or Rcp Transfer

The nature of the Internet being one of unpredictable performance at times, occasionally you want to know if a file transfer, being done by ftp or rcp, is making any progress.

To use lsof for watching a file transfer, you need to know the PID of the file transfer process. You can use ps to find that. Then use lsof,

```
$ lsof -p<PID>
```

to examine the files open to the transfer process. Usually the ftp files of interest are at file descriptors 9 and 10 or 10 and 11; for rcp, 3 and 4. They describe the network socket file and the local data file.

If you want to watch only those file descriptors as the file transfer progresses, try these lsof forms (for ftp in the example):

```
$ lsof -p<PID> -ad9,10 -r or $ lsof -p<PID> -ad10,11 -r
```

Some options need explaining:

**-p<PID>** specifies that lsof is to restrict its attention to the process whose ID is <PID>. You can specify a set of PIDs by separating them with commas.

```
$ lsof -p 1234,5678,9012
```

**-a** specifies that lsof is to AND its tests together. The two tests that are specified are tests on the PID and tests on file descriptions ("d9,10").

**d9,10** specifies that lsof is to test only file descriptors 9 and 10. Note that the '-' is absent, since "-a" is a unary option and can be followed immediately by another lsof option.

**-r** tells lsof to list the requested open file information, sleep for a default 15 seconds, then list the open file information again. You can specify a different time (in seconds) after -r and override the default.

# A Quick Start for lsof

Vic Abell

Lsof issues a short line of equal signs between each set of output to distinguish it.

For an rcp transfer, the above example becomes:

```
$ lsof -p<PID> -ad3,4 -r
```

## Listing Open NFS Files

Lsof will list all files open on remote file systems, supported by an NFS server. Just use:

```
$ lsof -N
```

Note, however, that when run on an NFS server, lsof will not list files open to the server from one of its clients. That's because lsof can only examine the processes running on the machine where it is called -- i.e., on the NFS server.

If you run lsof on the NFS client, using the -N option, it will list files open by processes on the client that are on remote NFS file systems.

## Listing Files Open by a Specific Login

If you're interested in knowing what files the processes owned by a particular login name have open, lsof can help.

```
$ lsof -u<login> or $ lsof -u<User ID number>
```

You can specify either the login name or the UID associated with it. You can specify multiple login names and UID numbers, mixed together, by separating them with commas.

```
$ lsof -u548,abe
```

On the subject of login names and UIDs, it's worth noting that lsof can be told to report either. By default it reports login names; the -l option switches reporting to UIDs. You might want to use -l if login name lookup is slow for some reason.

## Ignoring a Specific Login

The -u option can also be used to direct lsof to ignore a specific login name or UID, or a list of them. Simply prefix the login names or UIDs with a '^' character, as you might do in a regular expression. The '^' prefix is useful, for example, when you want to have lsof ignore the files open to system processes, owned by the root (UID 0) login. Try:

```
$ lsof -u ^root or $ lsof -u ^0
```

## Listing Files Open to a Specific Process Group

There's a Unix collection of processes called a process group. The name indicates that the processes of the group have a common association and are grouped so that a signal sent to one (e.g., a keyboard kill stroke) is delivered to all.

This causes Unix to create a two element process group:

```
$ lsof | less
```

# A Quick Start for lsof

Vic Abell

You can use lsof to look at the open files of all members of a process group, if you know the process group ID number. Assuming that it is 26168 for the above example, this lsof command:

```
$ lsof -g26168 -adcwd
```

would produce on a SunOS 4.1.3 system:

```
COMMAND  PID  PGRP  USER  FD  TYPE  DEVICE  SIZE/OFF  INODE  NAME
lsof     26168 26168  abe  cwd  VDIR   7, 30    1024 64372
/home/ipscgate/u11/abe/src/lsof4
less    26169 26168  abe  cwd  VDIR   7, 30    1024 64372
/home/ipscgate/u11/abe/src/lsof4
```

The ``-g26168" option specifies the process group ID of interest; the ``-adcwd" option specifies that options are to be ANDed and that lsof should limit file output to information about current working directory (``cwd") files.

## When Lsof Seems to Hang

On occasion when you run lsof it seems to hang and produce no output. This may result from system conditions beyond the control of lsof. lsof has a number of options that may allow you to bypass the blockage.

## Kernel lstat(), readlink(), and stat() Blockages

lsof uses the kernel (system) calls lstat(), readlink(), and stat() to locate mounted file system information. When a file system has been mounted from an NFS server and that server is temporarily unavailable, the calls lsof uses may block in the kernel.

lsof will announce that it is being blocked with warning messages (unless they have been suppressed by the lsof builder), but only after a default waiting period of fifteen seconds has expired for each file system whose server is unavailable. If you have a number of such file systems, the total wait may be unacceptably long.

You can do two things to shorten your suffering: 1) reduce the wait time with the -S option; or 2) tell lsof to avoid the kernel calls that might block by specifying the -b option.

```
$ lsof -S 5 or $ lsof -b
```

Avoiding the kernel calls that might block may result in the lack of some information that lsof needs to know about mounted file systems. Thus, when you use -b, lsof warns that it might lack important information.

The warnings that result from using -b (unless suppressed by the lsof builder) can themselves be annoying. You can suppress them by adding the -w option. (Of course, if you do, you won't know what warning messages lsof might have issued.)

```
$ lsof -bw
```

Note: if the lsof builder suppressed warning message issuance, you don't need to use -w to suppress them. You can tell what the default state of message warning issuance is by looking at the -h (help) output. If it says ``-w enable warnings" then warnings are disabled by default; ``-w disable warnings", they are enabled by default.

# A Quick Start for lsof

Vic Abell

## Problems with /dev or /devices

Lsof scans the /dev or /devices branch of your file system to obtain information about your system's devices. (The scan isn't necessary when a device cache file exists.)

Sometimes that scan can take a very long time, especially if you have a large number of devices, and if your kernel is relatively slow to process the stat() system call on device nodes. You can't do anything about the stat() system call speed.

However, you can make sure that lsof is allowed to use its device cache file feature. When lsof can use a device cache file, it retains information it gleans via the stat() calls on /dev or /devices in a separate file for later, faster access.

The device cache file feature is described in the lsof man page. See the DEVICE CACHE FILE, LSOFF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS, DEVICE CACHE FILE PATH FROM THE -D OPTION, DEVICE CACHE PATH FROM AN ENVIRONMENT VARIABLE, SYSTEM-WIDE DEVICE CACHE PATH, PERSONAL DEVICE CACHE PATH (DEFAULT), and MODIFIED PERSONAL DEVICE CACHE PATH sections.

There is also a separate file in the lsof distribution, named 00DCACHE, that describes the device cache file in detail, including information about possible security problems.

One final observation: don't overlook the possibility that your /dev or /devices tree might be damaged. See if

```
$ ls -R /dev or $ ls -R /devices
```

completes or hangs. If it hangs, then lsof will probably hang, too, and you should try to discover why ls hangs.

## Host and Service Name Lookup Hangs

Lsof can hang up when it tries to convert an Internet dot-form address to a host name, or a port number to a service name. Both hangs are caused by the lookup functions of your system.

An independent check for both types of hangs can be made with the netstat program. Run it without arguments. If it hangs, then it is probably having lookup difficulties. When you run it with -n it shouldn't hang and should report network and port numbers instead of names.

Lsof has two options that serve the same purpose as netstat's -n option. The lsof -n option tells it to avoid host name lookups; and -P, service name lookups. Try those options when you suspect lsof may be hanging because of lookup problems.

```
$ lsof -n or $ lsof -P or $ lsof -nP
```

## UID to Login Name Conversion Delays

By default lsof converts User IDentification (UID) numbers to login names when it produces output. That conversion process may sometimes hang because of system problems or interlocks.

You can tell lsof to skip the lookup with the -l option; it will then report UIDs in the USER column.

```
$ lsof -l
```

## Output for Other Programs

# A Quick Start for lsof

Vic Abell

The -F option allows you to specify that lsof should describe open files with a special form of output, called field output, that can be parsed easily by a subsequent program. The lsof distribution comes with sample AWK, Perl 4, and Perl 5 scripts that post-process field output.

The lsof manual page describes field output in detail in its OUTPUT FOR OTHER PROGRAMS section. A quick look at a sample script in the scripts/ subdirectory of the lsof distribution will also give you an idea how field output works.

The most important thing about field output is that it is relatively homogeneous across Unix dialects. Thus, if you write a script to post-process field output for AIX, it probably will work for HP-UX, Solaris, and Ultrix as well.

## The lsof Exit Code and Shell Scripts

When lsof exits successfully it returns an exit code based on the result of its search for specified files. (If no files were specified, then the successful exit code is 0 (zero).)

If lsof was asked to search for specific files, including any files on specified file systems, it returns an exit code of 0 (zero) if it found all the specified files and at least one file on each specified file system. Otherwise it returns a 1 (one).

If lsof detects an error and makes an unsuccessful exit, it returns an exit code of 1 (one).

You can use the exit code in a shell script to search for files on a file system and take action based on the result -- e.g.,

```
#!/bin/sh
lsof <file_system_name> > /dev/null 2>&1
if test $? -eq 0
then
echo "<file_system_name> may have no users."
else
echo "<file_system_name> has some users."
fi
```

## Options

The following appendices describe the lsof options in detail.

### Selection Options

lsof has a rich set of options for selecting the files to be displayed. These include:

- a tells lsof to AND the set of selection options that are specified. Normally lsof ORs them.

For example, if you specify the -p<PID> and -u<UID> options, lsof will display all files for the specified PID or for the specified UID.

By adding -a, you specify that the listed files should be limited to PIDs owned by the specified UIDs -- i.e., they match the PIDs \*and\* the UIDs.

```
$ lsof -p1234 -au 5678
```

# A Quick Start for lsof

Vic Abell

- c** specifies that lsof should list files belonging to processes having the associated command name.

Hint: if you want to select files based on more than one command name, use multiple `-c<name>` specifications.

## \$ lsof -cldsof -cksh

- d** tells lsof to select by the associated file descriptor (FD) set. An FD set is a comma-separated list of numbers and the names lsof normally displays in its FD column: `cwd`, `Lnn`, `ltx`, `<number>`, etc. See the OUTPUT section of the lsof man page for the complete list of possible file descriptors. Example:

```
$ lsof -dcwd,0,1,2
```

- g** tells lsof to select by the associated process group ID (PGRP) set. The PGRP set is a comma-separated list of PGRP numbers. When `-g` is specified, it also enables the display of PGRP numbers.

Note: when `-g` isn't followed by a PGRP set, it simply selects the listing of PGRP for all processes. Examples:

```
$ lsof -g
$ lsof -g1234,5678
```

- i** tells lsof to display Internet socket files. If no protocol/address/port specification follows `-i`, lsof lists all Internet socket files.

If a specification follows `-i`, lsof lists only the socket files whose Internet addresses match the specification.

Hint: multiple addresses may be specified with multiple `-i` options. Examples:

```
$ lsof -iTCP
$ lsof -i@vic.cc.purdue.edu:sendmail
```

- N** selects the listing of files mounted on NFS devices.
- U** selects the listing of socket files in the Unix domain.

## Output Options

Lsof has these options to control its output format:

# A Quick Start for lsof

Vic Abell

- F produce output that can be parsed by a subsequent program.
- g print process group (PGRP) IDs.
- l list UID numbers instead of login names.
- n list network numbers instead of host names.
- o always list file offset.
- P list port numbers instead of port service names.
- s always list file size.

## Precautionary Options

Lsof uses system functions that can block or take a long time, depending on the health of the Unix dialect supporting it. These include:

- b directs lsof to avoid system functions -- e.g., lstat(2), readlink(2), stat(2) -- that might block in the kernel. See the BLOCKS AND TIMEOUTS section of the lsof man page.  
  
You might want to use this option when you have a mount from an NFS server that is not responding.
- C tells lsof to ignore the kernel's name cache. As a precaution this option will have little effect on lsof performance, but might be useful if the kernel's name cache is scrambled. (I've never seen that happen.)
- D might be used to direct lsof to ignore an existing device cache file and generate a new one from /dev (and /devices). This might be useful if you have doubts about the integrity of an existing device cache file.
- l tells lsof to list UID numbers instead of login names -- this is useful when UID to login name conversion is slow or inoperative.
- n tells lsof to avoid converting Internet addresses to host numbers. This might be useful when your host name lookup (e.g., DNS) is inoperative.
- O tells lsof to avoid its strategy of forking to perform potentially blocking kernel operations. While the forking allows lsof to detect that a block has occurred (and possibly break it), the

## A Quick Start for lsof

Vic Abell

fork operation is a costly one. Use the -O option with care, lest your lsof be blocked.

- P directs lsof to list port numbers instead of trying to convert them to port service names. This might be useful if port to service name lookups (e.g., via NIS) are slow or failing.
- S can be used to change the lstat/readlink/stat timeout interval that governs how long lsof waits for response from the kernel. This might be useful when an NFS server is slow or unresponsive. When lsof times out of a kernel function, it may have less information to display. Example:

```
$ lsof -S2
```

- w tells lsof to avoid issuing warning messages, if they are enabled by default, or enable them if they are disabled by default. Check the -h (help) output to determine their status. If it says ``-w enable warnings'', then warning messages are disabled by default; ``-w disable warnings'', they are enabled by default.

This may be a useful option, for example, when you specify -b, if warning messages are enabled, because it will suppress the warning messages lsof issues about avoiding functions that might block in the kernel.

### Miscellaneous lsof Options

There are some lsof options that are hard to classify, including:

- ? these options select help output.
- h
- F selects field output. Field output is a mode where lsof produces output that can be parsed easily by subsequent programs -- e.g., AWK or Perl scripts. See ``15. Output for Other Programs'' for more information.
- k specifies an alternate kernel symbol file -- i.e., where nlist() will get its information. Example:  

```
$ lsof -k/usr/crash/vmunix.1
```
- m specifies an alternate kernel memory file from which lsof will read kernel structures in place of /dev/kmem or kvm\_read(). Example:

## A Quick Start for lsof

Vic Abell

```
$ lsof -m/usr/crash/vmcore.n
```

- r tells lsof to repeat its scan every 15 seconds (the default when no associated value is specified). A repeat time, different from the default, can follow -r. Example:

```
$ lsof -r30
```

- v displays information about the building of the lsof executable.
- The double minus sign option may be used to signal the end of options. It's particularly useful when arguments to the last option are optional and you want to supply a file path that could be confused for arguments to the last option. Example:

```
$ lsof -g -- 1
```

Where `1' is a file path, not PGRP ID 1.