NAME

lsof - list open files

SYNOPSIS

lsof [**-?abChlnNOPQRtUvVX**] [**-A** *A*] [**-c** *c*] [**+c** *c*] [**+|-d** *d*] [**+|-D** *D*] [**+|-e** *s*] [**+|-E**] [**+|-f** **[cfgGn]**] [**-F** *[f]*] [**-g** *[s]*] [**-i** *[i]*] [**-k** *k*] [**-K** *k*] [**+|-L** *[l]*] [**+|-m** *m*] [**+|-M**] [**-o** *[o]*] [**-p** *s*] [**+|-r** *[t[m<fmt>]]*] [**-s** *[p:s]*] [**-S** *[t]*] [**-T** *[t]*] [**-u** *s*] [**+|-w**] [**-x** *[fl]*] [**-z** *[z]*] [**-Z** *[Z]*] [**--**] [*names*]

DESCRIPTION

Lsof revision 4.97.0 lists on its standard output file information about files opened by processes for the following UNIX dialects:

Apple Darwin 9 and Mac OS X 10.[567]
FreeBSD 8.[234], 9.0 and 1[012].0 for AMD64-based systems
Linux 2.1.72 and above for x86-based systems
Solaris 9, 10 and 11

(See the **DISTRIBUTION** section of this manual page for information on how to obtain the latest *lsof* revision.)

An open file may be a regular file, a directory, a block special file, a character special file, an executing text reference, a library, a stream or a network file (Internet socket, NFS file or UNIX domain socket.) A specific file or all the files in a file system may be selected by path.

Instead of a formatted display, *lsof* will produce output that can be parsed by other programs. See the **-F**, option description, and the **OUTPUT FOR OTHER PROGRAMS** section for more information.

In addition to producing a single output list, *lsof* will run in repeat mode. In repeat mode it will produce output, delay, then repeat the output operation until stopped with an interrupt or quit signal. See the **+|-r** *[t[m<fmt>]]* option description for more information.

OPTIONS

In the absence of any options, *lsof* lists all open files belonging to all active processes.

If any list request option is specified, other list requests must be specifically requested - e.g., if **-U** is specified for the listing of UNIX socket files, NFS files won't be listed unless **-N** is also specified; or if a user list is specified with the **-u** option, UNIX domain socket files, belonging to users not in the list, won't be listed unless the **-U** option is also specified.

Normally, list options that are specifically stated are ORed - i.e., specifying the **-i** option without an

address and the **-u**foo option produces a listing of all network files OR files belonging to processes owned by user ''foo''. The exceptions are:

1) the '^' (negated) login name or user ID (UID), specified with the **-u** option;

2) the '^' (negated) process ID (PID), specified with the **-p** option;

3) the '^' (negated) process group ID (PGID), specified with the **-g** option;

4) the '^' (negated) command, specified with the **-c** option;

5) the ('^') negated TCP or UDP protocol state names, specified with the **-s** *[p:s]* option.

Since they represent exclusions, they are applied without ORing or ANDing and take effect before any other selection criteria are applied.

The **-a** option may be used to AND the selections. For example, specifying **-a**, **-U**, and **-u**foo produces a listing of only UNIX socket files that belong to processes owned by user ''foo''.

Caution: the **-a** option causes all list selection options to be ANDed; it can't be used to cause ANDing of selected pairs of selection options by placing it between them, even though its placement there is acceptable. Wherever **-a** is placed, it causes the ANDing of all selection options.

Items of the same selection set - command names, file descriptors, network addresses, process identifiers, user identifiers, zone names, security contexts - are joined in a single ORed set and applied before the result participates in ANDing. Thus, for example, specifying **-i**@aaa.bbb, **-i**@ccc.ddd, **-a**, and **-u**fff,ggg will select the listing of files that belong to either login ''fff'' OR ''ggg'' AND have network connections to either host aaa.bbb OR ccc.ddd.

Options may be grouped together following a single prefix -- e.g., the option set ''**-a -b -C**'' may be stated as **-abC**. However, since values are optional following **+|-f**, **-F**, **-g**, **-i**, **+|-L**, **-o**, **+|-r**, **-s**, **-S**, **-T**, **-x** and **-z**. when you have no values for them be careful that the following character isn't ambiguous. For example, **-Fn** might represent the **-F** and **-n** options, or it might represent the **n** field identifier character following the **-F** option. When ambiguity is possible, start a new option with a '-' character - e.g., ''**-F -n**''. If the next option is a file name, follow the possibly ambiguous option with ''--'' - e.g., ''**-F --** *name*''.

Either the '+' or the '-' prefix may be applied to a group of options. Options that don't take on separate meanings for each prefix - e.g., **-i** - may be grouped under either prefix. Thus, for example, ''+M -i'' may be stated as ''+Mi'' and the group means the same as the separate options. Be careful of prefix

grouping when one or more options in the group does take on separate meanings under different prefixes - e.g., **+|-M**; ''-iM'' is not the same request as ''-i +M''. When in doubt, use separate options with appropriate prefixes.

-? -h These two equivalent options select a usage (help) output list. *Lsof* displays a shortened form of this output when it detects an error in the options supplied to it, after it has displayed messages explaining each error. (Escape the '?' character as your shell requires.)

-a causes list selection options to be ANDed, as described above.

-A *A* is available on systems configured for AFS whose AFS kernel code is implemented via dynamic modules. It allows the *lsof* user to specify *A* as an alternate name list file where the kernel addresses of the dynamic modules might be found. See the *lsof* FAQ (The **FAQ** section gives its location.) for more information about dynamic modules, their symbols, and how they affect *lsof*.

-b causes *lsof* to avoid kernel functions that might block - *lstat*(2), *readlink*(2), and *stat*(2).

See the **BLOCKS AND TIMEOUTS** and **AVOIDING KERNEL BLOCKS** sections for information on using this option.

-c *c* selects the listing of files for processes executing the command that begins with the characters of *c*. Multiple commands may be specified, using multiple **-c** options. They are joined in a single ORed set before participating in AND option selection.

If *c* begins with a '^', then the following characters specify a command name whose processes are to be ignored (excluded.)

If *c* begins and ends with a slash ('/'), the characters between the slashes are interpreted as a regular expression. Shell meta-characters in the regular expression must be quoted to prevent their interpretation by the shell. The closing slash may be followed by these modifiers:

b	the regular expression is a basic one.
i	ignore the case of letters.
x	the regular expression is an extended one (default).

See the *lsof* FAQ (The **FAQ** section gives its location.) for more information on basic and extended regular expressions.

The simple command specification is tested first. If that test fails, the command regular expression is applied. If the simple command test succeeds, the command regular expression test isn't made. This may result in ''no command found for regex:'' messages when lsof's **-V** option is specified.

+c *w* defines the maximum number of initial characters of the name, supplied by the UNIX dialect, of the UNIX command associated with a process to be printed in the COMMAND column. (The *lsof* default is nine.)

Note that many UNIX dialects do not supply all command name characters to *lsof* in the files and structures from which *lsof* obtains command name. Often dialects limit the number of characters supplied in those sources. For example, Linux 2.4.27 and Solaris 9 both limit command name length to 16 characters.

If *w* is zero ('0'), all command characters supplied to *lsof* by the UNIX dialect will be printed.

If *w* is less than the length of the column title, ''COMMAND'', it will be raised to that length.

-C disables the reporting of any path name components from the kernel's name cache. See the **KERNEL NAME CACHE** section for more information.

+d *s* causes *lsof* to search for all open instances of directory *s* and the files and directories it contains at its top level. **+d** does NOT descend the directory tree, rooted at *s*. The **+D** *D* option may be used to request a full-descent directory tree search, rooted at directory *D*.

Processing of the **+d** option does not follow symbolic links within *s* unless the **-x** or **-x l** option is also specified. Nor does it search for open files on file system mount points on subdirectories of *s* unless the **-x** or **-x f** option is also specified.

Note: the authority of the user of this option limits it to searching for files that the user has permission to examine with the system *stat*(2) function.

-d *s* specifies a list of file descriptors (FDs) to exclude from or include in the output listing. The file descriptors are specified in the comma-separated set *s* - e.g., ''cwd,1,3'', ''^6,^2''. (There should be no spaces in the set.)

The list is an exclusion list if all entries of the set begin with '^'. It is an inclusion list if no entry begins with '^'. Mixed lists are not permitted.

A file descriptor number range may be in the set as long as neither member is empty, both

members are numbers, and the ending member is larger than the starting one - e.g., ''0-7'' or ''3-10''. Ranges may be specified for exclusion if they have the '^' prefix - e.g., ''^0-7'' excludes all file descriptors 0 through 7.

Multiple file descriptor numbers are joined in a single ORed set before participating in AND option selection.

When there are exclusion and inclusion members in the set, *lsof* reports them as errors and exits with a non-zero return code.

See the description of File Descriptor (FD) output values in the **OUTPUT** section for more information on file descriptor names.

fd is a pseudo file descriptor name for specifying the whole range of possible file descriptor numbers. **fd** does not appear in FD column of output.

+D *D* causes *lsof* to search for all open instances of directory *D* and all the files and directories it contains to its complete depth.

Processing of the **+D** option does not follow symbolic links within *D* unless the **-x** or **-x l** option is also specified. Nor does it search for open files on file system mount points on subdirectories of *D* unless the **-x** or **-x f** option is also specified.

Note: the authority of the user of this option limits it to searching for files that the user has permission to examine with the system *stat*(2) function.

Further note: *lsof* may process this option slowly and require a large amount of dynamic memory to do it. This is because it must descend the entire directory tree, rooted at *D*, calling *stat*(2) for each file and directory, building a list of all the files it finds, and searching that list for a match with every open file. When directory *D* is large, these steps can take a long time, so use this option prudently.

-D *D* directs *lsof*'*s* use of the device cache file. The use of this option is sometimes restricted. See the **DEVICE CACHE FILE** section and the sections that follow it for more information on this option.

-D must be followed by a function letter; the function letter may optionally be followed by a path name. *Lsof* recognizes these function letters:

? - report device cache file paths

b - build the device cache file
i - ignore the device cache file
r - read the device cache file
u - read and update the device cache file

The **b**, **r**, and **u** functions, accompanied by a path name, are sometimes restricted. When these functions are restricted, they will not appear in the description of the **-D** option that accompanies **-h** or **-?** option output. See the **DEVICE CACHE FILE** section and the sections that follow it for more information on these functions and when they're restricted.

The **?** function reports the read-only and write paths that lsof can use for the device cache file, the names of any environment variables whose values *lsof* will examine when forming the device cache file path, and the format for the personal device cache file path. (Escape the '?' character as your shell requires.)

When available, the **b**, **r**, and **u** functions may be followed by the device cache file's path. The standard default is *.lsof_hostname* in the home directory of the real user ID that executes *lsof*, but this could have been changed when *lsof* was configured and compiled. (The output of the **-h** and **-?** options show the current default prefix - e.g., ''.lsof''.) The suffix, *hostname*, is the first component of the host's name returned by *gethostname*(2).

When available, the **b** function directs *lsof* to build a new device cache file at the default or specified path.

The **i** function directs *lsof* to ignore the default device cache file and obtain its information about devices via direct calls to the kernel.

The **r** function directs *lsof* to read the device cache at the default or specified path, but prevents it from creating a new device cache file when none exists or the existing one is improperly structured. The **r** function, when specified without a path name, prevents *lsof* from updating an incorrect or outdated device cache file, or creating a new one in its place. The **r** function is always available when it is specified without a path name argument; it may be restricted by the permissions of the *lsof* process.

When available, the **u** function directs *lsof* to read the device cache file at the default or specified path, if possible, and to rebuild it, if necessary. This is the default device cache file function when no **-D** option has been specified.

+|-e *s* exempts the file system whose path name is *s* from being subjected to kernel function calls that might block. The **+e** option exempts *stat*(2), *lstat*(2) and most *readlink*(2) kernel function calls.

The **-e** option exempts only *stat(2)* and *lstat*(2) kernel function calls. Multiple file systems may be specified with separate **+|-e** specifications and each may have *readlink*(2) calls exempted or not.

This option is currently implemented only for Linux.

CAUTION: this option can easily be mis-applied to other than the file system of interest, because it uses path name rather than the more reliable device and inode numbers. (Device and inode numbers are acquired via the potentially blocking *stat*(2) kernel call and are thus not available, but see the **+|-m** *m* option as a possible alternative way to supply device numbers.) **Use this option with great care and fully specify the path name of the file system to be exempted.**

When open files on exempted file systems are reported, it may not be possible to obtain all their information. Therefore, some information columns will be blank, the characters ''UNKN'' preface the values in the TYPE column, and the applicable exemption option is added in parentheses to the end of the NAME column. (Some device number information might be made available via the **+|-m** *m* option.)

+|-E **+E** specifies that Linux pipe, Linux UNIX socket, Linux INET(6) socket closed in a local host, Linux pseudoterminal files, POSIX Message Queueue implementation in Linux, and Linux eventfd should be displayed with endpoint information and the files of the endpoints should also be displayed.

Note 1: UNIX socket file endpoint information is only available when the compile flags line of **-v** output contains HASUXSOCKEPT, and psudoterminal endpoint information is only available when the compile flags line contains HASPTYEPT.

Note 2: POSIX Message Queue file endpoint information is only available when mqueue file system is mounted.

Pipe endpoint information is displayed in the NAME column in the form ''*PID,cmd,FDmode*'', where *PID* is the endpoint process ID; *cmd* is the endpoint process command; *FD* is the endpoint file's descriptor; and *mode* is the endpoint file's access mode.

Pseudoterminal endpoint information is displayed in the NAME column as ''->/dev/pts*min PID,cmd,FDmode*'' or ''*PID,cmd,FDmode*''. The first form is for a master device; the second, for a slave device. *min* is a slave device's minor device number; and *PID, cmd, FD* and *mode* are the same as with pipe endpoint information. Note: psudoterminal endpoint information is only available when the compile flags line of **-V** output contains

HASPTYEPT. In addition, this feature works on Linux kernels above 4.13.0.

UNIX socket file endpoint information is displayed in the NAME column in the form ''type=*TYPE* ->INO=*INODE PID,cmd,FDmode*'', where *TYPE* is the socket type; *INODE* is the i-node number of the connected socket; and *PID, cmd, FD* and *mode* are the same as with pipe endpoint information. Note: UNIX socket file endpoint information is available only when the compile flags line of **-v** output contains HASUXSOCKEPT.

INET socket file endpoint information is inserted to the value at the NAME column in th form *PID, cmd, FD* and *mode* are the same as with pipe endpoint information. The endpoint information is available only if the socket is used for local IPC; both endpoints bind to the same local IPv4 or IPv6 address.

POSIX Message Queue file endpoint information is displayed in the NAME column in the same form as that of pipe.

eventfd endpoint information is displayed in the NAME column in the same form as that of pipe. This feature works on Linux kernels above 5.2.0.

Multiple occurrences of this information can appear in a file's NAME column.

-E specifies that endpoint supported files should be displayed with endpoint information, but not the files of the endpoints.

+|-f [cfgGn]

f by itself clarifies how path name arguments are to be interpreted. When followed by **c**, **f**, **g**, **G**, or **n** in any combination it specifies that the listing of kernel file structure information is to be enabled ('+') or inhibited ('-').

Normally a path name argument is taken to be a file system name if it matches a mounted-on directory name reported by *mount*(8), or if it represents a block device, named in the *mount* output and associated with a mounted directory name. When **+f** is specified, all path name arguments will be taken to be file system names, and *lsof* will complain if any are not. This can be useful, for example, when the file system name (mounted-on device) isn't a block device. This happens for some CD-ROM file systems.

When **-f** is specified by itself, all path name arguments will be taken to be simple files. Thus, for example, the ''**-f** -- /'' arguments direct lsof to search for open files with a '/' path name, not all open files in the '/' (root) file system.

Be careful to make sure **+f** and **-f** are properly terminated and aren't followed by a character (e.g., of the file or file system name) that might be taken as a parameter. For example, use ''--'' after **+f** and **-f** as in these examples.

```
$ lsof +f -- /file/system/name
$ lsof -f -- /file/name
```

The listing of information from kernel file structures, requested with the **+f [cfgGn]** option form, is normally inhibited, and is not available in whole or part for some dialects - e.g., /proc-based Linux kernels below 2.6.22. When the prefix to **f** is a plus sign ('+'), these characters request file structure information:

c file structure use count (not Linux)
f file structure address (not Linux)
g file flag abbreviations (Linux 2.6.22 and up)

Abbrev.Flag in C code (see open(2))

W	O_WRONLY
RW	O_RDWR
CR	O_CREAT
EXCL	O_EXCL
NTTY	O_NOCTTY
TR	O_TRUNC
AP	O_APPEND
ND	O_NDELAY
SYN	O_SYNC
ASYN	O_ASYNC
DIR	O_DIRECT
DTY	O_DIRECTORY
NFLK	O_NOFOLLOW
NATM	O_NOATIME
DSYN	O_DSYNC
RSYN	O_RSYNC
LG	O_LARGEFILE
CX	O_CLOEXEC
TMPF	O_TMPFILE

G file flags in hexadecimal (Linux 2.6.22 and up)
n file structure node address (not Linux)

When the prefix is minus ('-') the same characters disable the listing of the indicated values.

File structure addresses, use counts, flags, and node addresses may be used to detect more readily identical files inherited by child processes and identical files in use by different processes. *Lsof* column output can be sorted by output columns holding the values and listed to identify identical file use, or *lsof* field output can be parsed by an AWK or Perl post-filter script, or by a C program.

-F *f* specifies a character list, *f*, that selects the fields to be output for processing by another program, and the character that terminates each output field. Each field to be output is specified with a single character in *f*. The field terminator defaults to NL, but may be changed to NUL (000). See the **OUTPUT FOR OTHER PROGRAMS** section for a description of the field identification characters and the field output process.

When the field selection character list is empty, all standard fields are selected (except the raw device field, security context and zone field for compatibility reasons) and the NL field terminator is used.

When the field selection character list contains only a zero ('0'), all fields are selected (except the raw device field for compatibility reasons) and the NUL terminator character is used.

Other combinations of fields and their associated field terminator character must be set with explicit entries in *f*, as described in the **OUTPUT FOR OTHER PROGRAMS** section.

When a field selection character identifies an item *lsof* does not normally list - e.g., PPID, selected with **-R** - specification of the field character - e.g., ''**-FR**'' - also selects the listing of the item.

When the field selection character list contains the single character '?', *lsof* will display a help list of the field identification characters. (Escape the '?' character as your shell requires.)

-g *[s]* excludes or selects the listing of files for the processes whose optional process group IDentification (PGID) numbers are in the comma-separated set *s* - e.g., ''123'' or ''123,^456''. (There should be no spaces in the set.)

PGID numbers that begin with '^' (negation) represent exclusions.

Multiple PGID numbers are joined in a single ORed set before participating in AND option selection. However, PGID exclusions are applied without ORing or ANDing and take effect before other selection criteria are applied.

The **-g** option also enables the output display of PGID numbers. When specified without a PGID set that's all it does.

-i *[i]* selects the listing of files any of whose Internet address matches the address specified in *i*. If no address is specified, this option selects the listing of all Internet and x.25 (HP-UX) network files.

If **-i***4* or **-i***6* is specified with no following address, only files of the indicated IP version, IPv4 or IPv6, are displayed. (An IPv6 specification may be used only if the dialects supports IPv6, as indicated by ''[46]'' and ''IPv[46]'' in *lsof*'*s* **-h** or **-?** output.) Sequentially specifying **-i**4, followed by **-i**6 is the same as specifying **-i**, and vice-versa. Specifying **-i**4, or **-i**6 after **-i** is the same as specifying **-i**4 or **-i**6 by itself.

Multiple addresses (up to a limit of 100) may be specified with multiple **-i** options. (A port number or service name range is counted as one address.) They are joined in a single ORed set before participating in AND option selection.

An Internet address is specified in the form (Items in square brackets are optional.):

[*46*][*protocol*][*@hostname*|*hostaddr*][:*service*|*port*]

where:

46 specifies the IP version, IPv4 or IPv6 that applies to the following address. '6' may be be specified only if the UNIX dialect supports IPv6. If neither '4' nor '6' is specified, the following address applies to all IP versions.

protocol is a protocol name - **TCP**, **UDP** or **UDPLITE**.

hostname is an Internet host name. Unless a specific IP version is specified, open network files associated with host names of all versions will be selected.

hostaddr is a numeric Internet IPv4 address in dot form; or an IPv6 numeric address in colon form, enclosed in brackets, if the UNIX dialect supports IPv6. When an IP version is selected, only its numeric addresses may be specified.

service is an */etc/services* name - e.g., **smtp** -
or a list of them.
port is a port number, or a list of them.

IPv6 options may be used only if the UNIX dialect supports IPv6. To see if the dialect supports IPv6, run *lsof* and specify the **-h** or **-?** (help) option. If the displayed description of the **-i** option contains ''[46]'' and ''IPv[46]'', IPv6 is supported.

IPv4 host names and addresses may not be specified if network file selection is limited to IPv6 with **-i** 6. IPv6 host names and addresses may not be specified if network file selection is limited to IPv4 with **-i** 4. When an open IPv4 network file's address is mapped in an IPv6 address, the open file's type will be IPv6, not IPv4, and its display will be selected by '6', not '4'.

At least one address component - **4, 6,** *protocol*, *hostname*, *hostaddr*, or *service* - must be supplied. The '@' character, leading the host specification, is always required; as is the ':', leading the port specification. Specify either *hostname* or *hostaddr*. Specify either *service* name list or *port* number list. If a *service* name list is specified, the *protocol* may also need to be specified if the TCP, UDP and UDPLITE port numbers for the service name are different. Use any case - lower or upper - for *protocol*.

Service names and *port* numbers may be combined in a list whose entries are separated by commas and whose numeric range entries are separated by minus signs. There may be no embedded spaces, and all service names must belong to the specified *protocol*. Since service names may contain embedded minus signs, the starting entry of a range can't be a service name; it can be a port number, however.

Here are some sample addresses:

-i6 - IPv6 only
TCP:25 - TCP and port 25
@1.2.3.4 - Internet IPv4 host address 1.2.3.4
@[3ffe:1ebc::1]:1234 - Internet IPv6 host address
3ffe:1ebc::1, port 1234
UDP:who - UDP who service port
TCP@lsof.itap:513 - TCP, port 513 and host name lsof.itap
tcp@foo:1-10,smtp,99 - TCP, ports 1 through 10,
service name *smtp*, port 99, host name foo
tcp@bar:1-smtp - TCP, ports 1 through *smtp*, host bar
:time - either TCP, UDP or UDPLITE time service port

-K *k* selects the listing of tasks (threads) of processes, on dialects where task (thread) reporting is supported. (If help output - i.e., the output of the **-h** or **-?** options - shows this option, then task (thread) reporting is supported by the dialect.)

If **-K** is followed by a value, *k*, it must be ''i''. That causes *lsof* to ignore tasks, particularly in the default, list-everything case when no other options are specified.

When **-K** and **-a** are both specified on Linux, and the tasks of a main process are selected by other options, the main process will also be listed as though it were a task, but without a task ID. (See the description of the TID column in the **OUTPUT** section.)

Where the FreeBSD version supports threads, all threads will be listed with their IDs.

In general threads and tasks inherit the files of the caller, but may close some and open others, so *lsof* always reports all the open files of threads and tasks.

-k *k* specifies a kernel name list file, *k*, in place of /vmunix, /mach, etc. **-k** is not available under AIX on the IBM RISC/System 6000.

-l inhibits the conversion of user ID numbers to login names. It is also useful when login name lookup is working improperly or slowly.

+|-L *[l]* enables ('+') or disables ('-') the listing of file link counts, where they are available - e.g., they aren't available for sockets, or most FIFOs and pipes.

When **+L** is specified without a following number, all link counts will be listed. When **-L** is specified (the default), no link counts will be listed.

When **+L** is followed by a number, only files having a link count less than that number will be listed. (No number may follow **-L**.) A specification of the form ''**+L1**'' will select open files that have been unlinked. A specification of the form ''**+aL1** <*file_system*>'' will select unlinked open files on the specified file system.

For other link count comparisons, use field output (**-F**) and a post-processing script or program.

+|-m *m* specifies an alternate kernel memory file or activates mount table supplement processing.

The option form **-m** *m* specifies a kernel memory file, *m*, in place of */dev/kmem* or */dev/mem* - e.g., a crash dump file.

The option form **+m** requests that a mount supplement file be written to the standard output file. All other options are silently ignored.

There will be a line in the mount supplement file for each mounted file system, containing the mounted file system directory, followed by a single space, followed by the device number in hexadecimal "0x" format - e.g.,

/ 0x801

Lsof can use the mount supplement file to get device numbers for file systems when it can't get them via *stat*(2) or *lstat*(2).

The option form **+m** *m* identifies *m* as a mount supplement file.

Note: the **+m** and **+m** *m* options are not available for all supported dialects. Check the output of *lsof's* **-h** or **-?** options to see if the **+m** and **+m** *m* options are available.

+|-M Enables (+) or disables (-) the reporting of portmapper registrations for local TCP, UDP and UDPLITE ports, where port mapping is supported. (See the last paragraph of this option description for information about where portmapper registration reporting is supported.)

The default reporting mode is set by the *lsof* builder with the HASPMAPENABLED #define in the dialect's machine.h header file; *lsof* is distributed with the HASPMAPENABLED #define deactivated, so portmapper reporting is disabled by default and must be requested with **+M**. Specifying *lsof's* **-h** or **-?** option will report the default mode. Disabling portmapper registration when it is already disabled or enabling it when already enabled is acceptable. When portmapper registration reporting is enabled, *lsof* displays the portmapper registration (if any) for local TCP, UDP or UDPLITE ports in square brackets immediately following the port numbers or service names - e.g., ''':1234[name]''' or ''':name[100083]'''. The registration information may be a name or number, depending on what the registering program supplied to the portmapper when it registered the port.

When portmapper registration reporting is enabled, *lsof* may run a little more slowly or even become blocked when access to the portmapper becomes congested or stopped. Reverse the reporting mode to determine if portmapper registration reporting is slowing or blocking *lsof*.

For purposes of portmapper registration reporting *lsof* considers a TCP, UDP or UDPLITE port local if: it is found in the local part of its containing kernel structure; or if it is located in the foreign part of its containing kernel structure and the local and foreign Internet addresses are the same; or if it is located in the foreign part of its containing kernel structure and the

foreign Internet address is INADDR_LOOPBACK (127.0.0.1). This rule may make *lsof* ignore some foreign ports on machines with multiple interfaces when the foreign Internet address is on a different interface from the local one.

See the *lsof* FAQ (The **FAQ** section gives its location.) for further discussion of portmapper registration reporting issues.

Portmapper registration reporting is supported only on dialects that have RPC header files. (Some Linux distributions with GlibC 2.14 do not have them.) When portmapper registration reporting is supported, the **-h** or **-?** help output will show the **+|-M** option.

-n inhibits the conversion of network numbers to host names for network files. Inhibiting conversion may make *lsof* run faster. It is also useful when host name lookup is not working properly.

-N selects the listing of NFS files.

-o directs *lsof* to display file offset at all times. It causes the SIZE/OFF output column title to be changed to OFFSET. Note: on some UNIX dialects *lsof* can't obtain accurate or consistent file offset information from its kernel data sources, sometimes just for particular kinds of files (e.g., socket files.) Consult the *lsof* FAQ (The **FAQ** section gives its location.) for more information.

The **-o** and **-s** options are mutually exclusive; they can't both be specified. When neither is specified, *lsof* displays whatever value - size or offset - is appropriate and available for the type of the file.

-o *o* defines the number of decimal digits (*o*) to be printed after the ''0t'' for a file offset before the form is switched to ''0x...''. An *o* value of zero (unlimited) directs *lsof* to use the ''0t'' form for all offset output.

This option does NOT direct *lsof* to display offset at all times; specify **-o** (without a trailing number) to do that. **-o** *o* only specifies the number of digits after ''0t'' in either mixed size and offset or offset-only output. Thus, for example, to direct *lsof* to display offset at all times with a decimal digit count of 10, use:

```
-o -o 10
```

or

```
-oo10
```

The default number of digits allowed after ''0t'' is normally 8, but may have been changed by the lsof builder. Consult the description of the **-o** *o* option in the output of the **-h** or **-?** option to determine the default that is in effect.

-O directs *lsof* to bypass the strategy it uses to avoid being blocked by some kernel operations - i.e., doing them in forked child processes. See the **BLOCKS AND TIMEOUTS** and **AVOIDING KERNEL BLOCKS** sections for more information on kernel operations that may block *lsof*.

While use of this option will reduce *lsof* startup overhead, it may also cause *lsof* to hang when the kernel doesn't respond to a function. Use this option cautiously.

-p *s* excludes or selects the listing of files for the processes whose optional process IDentification (PID) numbers are in the comma-separated set *s* - e.g., ''123'' or ''123,^456''. (There should be no spaces in the set.)

PID numbers that begin with '^' (negation) represent exclusions.

Multiple process ID numbers are joined in a single ORed set before participating in AND option selection. However, PID exclusions are applied without ORing or ANDing and take effect before other selection criteria are applied.

-P inhibits the conversion of port numbers to port names for network files. Inhibiting the conversion may make *lsof* run a little faster. It is also useful when port name lookup is not working properly.

-Q ignore failed search terms. When *lsof* is told to search for users of a file, or for users of a device, or for a specific PID, or for certain protocols in use by that PID, and so on, *lsof* will return an error if any of the search results are empty. The **-Q** option will change this behavior so that *lsof* will instead return a successful exit code (0) even if any of the search results are empty. In addition, missing search terms will not be reported to stderr.

+|-r *[t[c<N>][m<fmt>]]*

puts *lsof* in repeat mode. There *lsof* lists open files as selected by other options, delays *t* seconds (default fifteen), then repeats the listing, delaying and listing repetitively until stopped by a condition defined by the prefix to the option.

If the prefix is a '-', repeat mode is endless. *Lsof* must be terminated with an interrupt or quit signal. 'c<N>' is for specifying the limits of repeating; if the number of iterations reaches at '<N>', *Lsof* stops itself.

If the prefix is '+', repeat mode will end the first cycle no open files are listed - and of course when *lsof* is stopped with an interrupt or quit signal. When repeat mode ends because no files are listed, the process exit code will be zero if any open files were ever listed; one, if none were ever listed.

Lsof marks the end of each listing: if field output is in progress (the **-F**, option has been specified), the default marker is 'm'; otherwise the default marker is ''========''. The marker is followed by a NL character.

The optional "m<fmt>" argument specifies a format for the marker line. The <fmt> characters following 'm' are interpreted as a format specification to the *strftime*(3) function, when both it and the *localtime*(3) function are available in the dialect's C library. Consult the *strftime*(3) documentation for what may appear in its format specification. Note that when field output is requested with the **-F** option, <fmt> cannot contain the NL format, ''%n''. Note also that when <fmt> contains spaces or other characters that affect the shell's interpretation of arguments, <fmt> must be quoted appropriately.

Repeat mode reduces *lsof* startup overhead, so it is more efficient to use this mode than to call *lsof* repetitively from a shell script, for example.

To use repeat mode most efficiently, accompany **+|-r** with specification of other *lsof* selection options, so the amount of kernel memory access *lsof* does will be kept to a minimum. Options that filter at the process level - e.g., **-c**, **-g**, **-p**, **-u** - are the most efficient selectors.

Repeat mode is useful when coupled with field output (see the **-F**, option description) and a supervising *awk* or *Perl* script, or a C program.

-R directs lsof to list the Parent Process IDentification number in the PPID column.

-s *[p:s]* **s** alone directs *lsof* to display file size at all times. It causes the SIZE/OFF output column title to be changed to SIZE. If the file does not have a size, nothing is displayed.

The optional **-s** *p:s* form is available only for selected dialects, and only when the **-h** or **-?** help output lists it.

When the optional form is available, the **s** may be followed by a protocol name (*p*), either TCP or UDP, a colon (':') and a comma-separated protocol state name list, the option causes open TCP and UDP files to be excluded if their state name(s) are in the list (*s*) preceded by a '^'; or included if their name(s) are not preceded by a '^'.

Dialects that support this option may support only one protocol. When an unsupported protocol is specified, a message will be displayed indicating state names for the protocol are unavailable.

When an inclusion list is defined, only network files with state names in the list will be present in the *lsof* output. Thus, specifying one state name means that only network files with that lone state name will be listed.

Case is unimportant in the protocol or state names, but there may be no spaces and the colon (':') separating the protocol name (*p*) and the state name list (*s*) is required.

If only TCP and UDP files are to be listed, as controlled by the specified exclusions and inclusions, the **-i** option must be specified, too. If only a single protocol's files are to be listed, add its name as an argument to the **-i** option.

For example, to list only network files with TCP state LISTEN, use:

```
-iTCP -sTCP:LISTEN
```

Or, for example, to list network files with all UDP states except Idle, use:

```
-iUDP -sUDP:^Idle
```

State names vary with UNIX dialects, so it's not possible to provide a complete list. Some common TCP state names are: CLOSED, IDLE, BOUND, LISTEN, ESTABLISHED, SYN_SENT, SYN_RCDV, ESTABLISHED, CLOSE_WAIT, FIN_WAIT1, CLOSING, LAST_ACK, FIN_WAIT_2, and TIME_WAIT. Two common UDP state names are Unbound and Idle.

See the *lsof* FAQ (The **FAQ** section gives its location.) for more information on how to use protocol state exclusion and inclusion, including examples.

The **-o** (without a following decimal digit count) and **-s** option (without a following protocol and state name list) are mutually exclusive; they can't both be specified. When neither is specified, *lsof* displays whatever value - size or offset - is appropriate and available for the type of file.

Since some types of files don't have true sizes - sockets, FIFOs, pipes, etc. - lsof displays for their sizes the content amounts in their associated kernel buffers, if possible.

-S *[t]* specifies an optional time-out seconds value for kernel functions - *lstat*(2), *readlink*(2), and *stat*(2) - that might otherwise deadlock. The minimum for *t* is two; the default, fifteen; when no value is specified, the default is used.

See the **BLOCKS AND TIMEOUTS** section for more information.

-T *[t]* controls the reporting of some TCP/TPI information, also reported by *netstat*(1), following the network addresses. In normal output the information appears in parentheses, each item except TCP or TPI state name identified by a keyword, followed by '=', separated from others by a single space:

<TCP or TPI state name>
QR=<read queue length>
QS=<send queue length>
SO=<socket options and values>
SS=<socket states>
TF=<TCP flags and values>
WR=<window read length>
WW=<window write length>

Not all values are reported for all UNIX dialects. Items values (when available) are reported after the item name and '='.

When the field output mode is in effect (See **OUTPUT FOR OTHER PROGRAMS**.) each item appears as a field with a 'T' leading character.

-T with no following key characters disables TCP/TPI information reporting.

-T with following characters selects the reporting of specific TCP/TPI information:

f selects reporting of socket options, states and values, and TCP flags and values.
q selects queue length reporting.
s selects connection state reporting.
w selects window size reporting.

Not all selections are enabled for some UNIX dialects. State may be selected for all dialects and is reported by default. The **-h** or **-?** help output for the **-T** option will show what selections may be used with the UNIX dialect.

When **-T** is used to select information - i.e., it is followed by one or more selection characters - the displaying of state is disabled by default, and it must be explicitly selected again in the characters following **-T**. (In effect, then, the default is equivalent to **-Ts**.) For example, if queue lengths and state are desired, use **-Tqs**.

Socket options, socket states, some socket values, TCP flags and one TCP value may be reported (when available in the UNIX dialect) in the form of the names that commonly appear after SO_, so_, SS_, TCP_ and TF_ in the dialect's header files - most often <sys/socket.h>, <sys/socketvar.h> and <netinet/tcp_var.h>. Consult those header files for the meaning of the flags, options, states and values.

''SO='' precedes socket options and values; ''SS='', socket states; and ''TF='', TCP flags and values.

If a flag or option has a value, the value will follow an '=' and the name -- e.g., ''SO=LINGER=5'', ''SO=QLIM=5'', ''TF=MSS=512''. The following seven values may be reported:

Name Reported	Description (Common Symbol)
KEEPALIVE	keep alive time (SO_KEEPALIVE)
LINGER	linger time (SO_LINGER)
MSS	maximum segment size (TCP_MAXSEG)
PQLEN	partial listen queue connections
QLEN	established listen queue connections
QLIM	established listen queue limit
RCVBUF	receive buffer length (SO_RCVBUF)
SNDBUF	send buffer length (SO_SNDBUF)

Details on what socket options and values, socket states, and TCP flags and values may be displayed for particular UNIX dialects may be found in the answer to the ''Why doesn't lsof report socket options, socket states, and TCP flags and values for my dialect?'' and ''Why doesn't lsof report the partial listen queue connection count for my dialect?'' questions in the *lsof* FAQ (The **FAQ** section gives its location.) On Linux this option also prints the state of UNIX domain sockets.

-t produce terse output comprising only process identifiers (without a header), so that it is easy to use programmatically. e.g.

```
# reload anything using old SSL
lsof -t /lib/*/libssl.so.* | xargs -r kill -HUP

# get list of processes and then iterate over them (Bash only)
mapfile -t pids < <(
   lsof -wt /var/log/your.log
)
for pid in "${pids[@]}" ; do
   your_command -p "$pid"
done
```

The **-t** option implies the **-w** option.

-u *s* selects the listing of files for the user whose login names or user ID numbers are in the comma-separated set *s* - e.g., ‘‘abe’’, or ‘‘548,root’’. (There should be no spaces in the set.)

Multiple login names or user ID numbers are joined in a single ORed set before participating in AND option selection.

If a login name or user ID is preceded by a ‘^’, it becomes a negation - i.e., files of processes owned by the login name or user ID will never be listed. A negated login name or user ID selection is neither ANDed nor ORed with other selections; it is applied before all other selections and absolutely excludes the listing of the files of the process. For example, to direct *lsof* to exclude the listing of files belonging to root processes, specify ‘‘-u^root’’ or ‘‘-u^0’’.

-U selects the listing of UNIX domain socket files.

-v selects the listing of *lsof* version information, including: revision number; when the *lsof* binary was constructed; who constructed the binary and where; the name of the compiler used to construct the *lsof binary;* the version number of the compiler when readily available; the compiler and loader flags used to construct the *lsof* binary; and system information, typically the output of *uname*’s **-a** option.

-V directs *lsof* to indicate the items it was asked to list and failed to find - command names, file names, Internet addresses or files, login names, NFS files, PIDs, PGIDs, and UIDs.

When other options are ANDed to search options, or compile-time options restrict the listing of some files, *lsof* may not report that it failed to find a search item when an ANDed option or compile-time option prevents the listing of the open file containing the located search item.

For example, ‘‘lsof -V -iTCP@foobar -a -d 999’’ may not report a failure to locate open files at ‘‘TCP@foobar’’ and may not list any, if none have a file descriptor number of 999. A similar situation arises when HASSECURITY and HASNOSOCKSECURITY are defined at compile time and they prevent the listing of open files.

+|-w Enables (+) or disables (-) the suppression of warning messages.

The *lsof* builder may choose to have warning messages disabled or enabled by default. The default warning message state is indicated in the output of the **-h** or **-?** option. Disabling warning messages when they are already disabled or enabling them when already enabled is acceptable.

The **-t** option implies the **-w** option.

-x *[fl]* may accompany the **+d** and **+D** options to direct their processing to cross over symbolic links and|or file system mount points encountered when scanning the directory (**+d**) or directory tree (**+D**).

If **-x** is specified by itself without a following parameter, cross-over processing of both symbolic links and file system mount points is enabled. Note that when **-x** is specified without a parameter, the next argument must begin with ’-’ or ’+’.

The optional ’f’ parameter enables file system mount point cross-over processing; ’l’, symbolic link cross-over processing.

The **-x** option may not be supplied without also supplying a **+d** or **+D** option.

-X This is a dialect-specific option.

AIX:

This IBM AIX RISC/System 6000 option requests the reporting of executed text file and shared library references.

WARNING: because this option uses the kernel readx() function, its use on a busy AIX system might cause an application process to hang so completely that it can neither be killed nor stopped. I have never seen this happen or had a report of its happening, but I think there is a remote possibility it could happen.

By default use of readx() is disabled. On AIX 5L and above *lsof* may need setuid-root permission to perform the actions this option requests.

The *lsof* builder may specify that the **-X** option be restricted to processes whose real UID is root. If that has been done, the **-X** option will not appear in the **-h** or **-?** help output unless the real UID of the *lsof* process is root. The default *lsof* distribution allows any UID to specify **-X,** so by default it will appear in the help output.

When AIX readx() use is disabled, *lsof* may not be able to report information for all text and loader file references, but it may also avoid exacerbating an AIX kernel directory search kernel error, known as the Stale Segment ID bug.

The readx() function, used by *lsof* or any other program to access some sections of kernel virtual memory, can trigger the Stale Segment ID bug. It can cause the kernel's dir_search() function to believe erroneously that part of an in-memory copy of a file system directory has been zeroed. Another application process, distinct from *lsof*, asking the kernel to search the directory - e.g., by using *open*(2) - can cause dir_search() to loop forever, thus hanging the application process.

Consult the *lsof* FAQ (The **FAQ** section gives its location.) and the *00README* file of the *lsof* distribution for a more complete description of the Stale Segment ID bug, its APAR, and methods for defining readx() use when compiling *lsof*.

Linux:

This Linux option requests that *lsof* skip the reporting of information on all open TCP, UDP and UDPLITE IPv4 and IPv6 files.

This Linux option is most useful when the system has an extremely large number of open TCP, UDP and UDPLITE files, the processing of whose information in the */proc/net/tcp** and */proc/net/udp** files would take *lsof* a long time, and whose reporting is not of interest.

Use this option with care and only when you are sure that the information you want *lsof* to display isn't associated with open TCP, UDP or UDPLITE socket files.

Solaris 10 and above:

This Solaris 10 and above option requests the reporting of cached paths for files that have been deleted - i.e., removed with *rm*(1) or *unlink*(2).

The cached path is followed by the string '' (deleted)'' to indicate that the path by which the file was opened has been deleted.

Because intervening changes made to the path - i.e., renames with *mv*(1) or *rename*(2) - are not recorded in the cached path, what *lsof* reports is only the path by which the file was opened,

not its possibly different final path.

-z *[z]* specifies how Solaris 10 and higher zone information is to be handled.

Without a following argument - e.g., NO *z* - the option specifies that zone names are to be listed in the ZONE output column.

The **-z** option may be followed by a zone name, **z**. That causes lsof to list only open files for processes in that zone. Multiple **-z** *z* option and argument pairs may be specified to form a list of named zones. Any open file of any process in any of the zones will be listed, subject to other conditions specified by other options and arguments.

-Z *[Z]* specifies how SELinux security contexts are to be handled. It and 'Z' field output character support are inhibited when SELinux is disabled in the running Linux kernel. See **OUTPUT FOR OTHER PROGRAMS** for more information on the 'Z' field output character.

Without a following argument - e.g., NO *Z* - the option specifies that security contexts are to be listed in the SECURITY-CONTEXT output column.

The **-Z** option may be followed by a wildcard security context name, **Z**. That causes lsof to list only open files for processes in that security context. Multiple **-Z** *Z* option and argument pairs may be specified to form a list of security contexts. Any open file of any process in any of the security contexts will be listed, subject to other conditions specified by other options and arguments. Note that *Z* can be A:B:C or *:B:C or A:B:* or *:*:C to match against the A:B:C context.

-- The double minus sign option is a marker that signals the end of the keyed options. It may be used, for example, when the first file name begins with a minus sign. It may also be used when the absence of a value for the last keyed option must be signified by the presence of a minus sign in the following option and before the start of the file names.

names These are path names of specific files to list. Symbolic links are resolved before use. The first name may be separated from the preceding options with the ''--'' option.

If a *name* is the mounted-on directory of a file system or the device of the file system, *lsof* will list all the files open on the file system. To be considered a file system, the *name* must match a mounted-on directory name in *mount*(8) output, or match the name of a block device associated with a mounted-on directory name. The **+|-f** option may be used to force *lsof* to consider a *name* a file system identifier (**+f**) or a simple file (**-f**).

If *name* is a path to a directory that is not the mounted-on directory name of a file system, it is treated just as a regular file is treated - i.e., its listing is restricted to processes that have it open as a file or as a process-specific directory, such as the root or current working directory. To request that *lsof* look for open files inside a directory name, use the **+d** *s* and **+D** *D* options.

If a *name* is the base name of a family of multiplexed files - e.g, AIX's */dev/pt[cs]* - *lsof* will list all the associated multiplexed files on the device that are open - e.g., */dev/pt[cs]/1*, */dev/pt[cs]/2*, etc.

If a *name* is a UNIX domain socket name, *lsof* will usually search for it by the characters of the name alone - exactly as it is specified and is recorded in the kernel socket structure. (See the next paragraph for an exception to that rule for Linux.) Specifying a relative path - e.g., *./file* - in place of the file's absolute path - e.g., */tmp/file* - won't work because *lsof* must match the characters you specify with what it finds in the kernel UNIX domain socket structures.

If a *name* is a Linux UNIX domain socket name, in one case *lsof* is able to search for it by its device and inode number, allowing *name* to be a relative path. The case requires that the absolute path -- i.e., one beginning with a slash ('/') be used by the process that created the socket, and hence be stored in the */proc/net/unix* file; and it requires that *lsof* be able to obtain the device and node numbers of both the absolute path in */proc/net/unix* and *name* via successful *stat*(2) system calls. When those conditions are met, *lsof* will be able to search for the UNIX domain socket when some path to it is is specified in *name*. Thus, for example, if the path is */dev/log*, and an *lsof* search is initiated when the working directory is */dev*, then *name* could be *./log*.

If a *name* is none of the above, *lsof* will list any open files whose device and inode match that of the specified path *name*.

If you have also specified the **-b** option, the only *names* you may safely specify are file systems for which your mount table supplies alternate device numbers. See the **AVOIDING KERNEL BLOCKS** and **ALTERNATE DEVICE NUMBERS** sections for more information.

Multiple file names are joined in a single ORed set before participating in AND option selection.

AFS

Lsof supports the recognition of AFS files for these dialects (and AFS versions):

AIX 4.1.4 (AFS 3.4a)
HP-UX 9.0.5 (AFS 3.4a)

Linux 1.2.13 (AFS 3.3)
Solaris 2.[56] (AFS 3.4a)

It may recognize AFS files on other versions of these dialects, but has not been tested there. Depending on how AFS is implemented, *lsof* may recognize AFS files in other dialects, or may have difficulties recognizing AFS files in the supported dialects.

Lsof may have trouble identifying all aspects of AFS files in supported dialects when AFS kernel support is implemented via dynamic modules whose addresses do not appear in the kernel's variable name list. In that case, *lsof* may have to guess at the identity of AFS files, and might not be able to obtain volume information from the kernel that is needed for calculating AFS volume node numbers. When *lsof* can't compute volume node numbers, it reports blank in the NODE column.

The **-A** *A* option is available in some dialect implementations of *lsof* for specifying the name list file where dynamic module kernel addresses may be found. When this option is available, it will be listed in the *lsof* help output, presented in response to the **-h** or **-?**

See the *lsof* FAQ (The **FAQ** section gives its location.) for more information about dynamic modules, their symbols, and how they affect *lsof* options.

Because AFS path lookups don't seem to participate in the kernel's name cache operations, *lsof* can't identify path name components for AFS files.

SECURITY

Lsof has three features that may cause security concerns. First, its default compilation mode allows anyone to list all open files with it. Second, by default it creates a user-readable and user-writable device cache file in the home directory of the real user ID that executes *lsof*. (The list-all-open-files and device cache features may be disabled when *lsof* is compiled.) Third, its **-k** and **-m** options name alternate kernel name list or memory files.

Restricting the listing of all open files is controlled by the compile-time HASSECURITY and HASNOSOCKSECURITY options. When HASSECURITY is defined, *lsof* will allow only the root user to list all open files. The non-root user may list only open files of processes with the same user IDentification number as the real user ID number of the *lsof* process (the one that its user logged on with).

However, if HASSECURITY and HASNOSOCKSECURITY are both defined, anyone may list open socket files, provided they are selected with the **-i** option.

When HASSECURITY is not defined, anyone may list all open files.

Help output, presented in response to the **-h** or **-?** option, gives the status of the HASSECURITY and HASNOSOCKSECURITY definitions.

See the **Security** section of the *00README* file of the *lsof* distribution for information on building *lsof* with the HASSECURITY and HASNOSOCKSECURITY options enabled.

Creation and use of a user-readable and user-writable device cache file is controlled by the compile-time HASDCACHE option. See the **DEVICE CACHE FILE** section and the sections that follow it for details on how its path is formed. For security considerations it is important to note that in the default *lsof* distribution, if the real user ID under which *lsof* is executed is root, the device cache file will be written in root's home directory - e.g., */* or */root*. When HASDCACHE is not defined, *lsof* does not write or attempt to read a device cache file.

When HASDCACHE is defined, the *lsof* help output, presented in response to the **-h**, **-D?**, or **-?** options, will provide device cache file handling information. When HASDCACHE is not defined, the **-h** or **-?** output will have no **-D** option description.

Before you decide to disable the device cache file feature - enabling it improves the performance of *lsof* by reducing the startup overhead of examining all the nodes in */dev* (or */devices*) - read the discussion of it in the *00DCACHE* file of the *lsof* distribution and the *lsof* FAQ (The **FAQ** section gives its location.)

WHEN IN DOUBT, YOU CAN TEMPORARILY DISABLE THE USE OF THE DEVICE CACHE FILE WITH THE **-Di** OPTION.

When *lsof* user declares alternate kernel name list or memory files with the **-k** and **-m** options, *lsof* checks the user's authority to read them with *access*(2). This is intended to prevent whatever special power *lsof's* modes might confer on it from letting it read files not normally accessible via the authority of the real user ID.

OUTPUT

This section describes the information *lsof* lists for each open file. See the **OUTPUT FOR OTHER PROGRAMS** section for additional information on output that can be processed by another program.

Lsof only outputs printable (declared so by *isprint*(3)) 8 bit characters. Non-printable characters are printed in one of three forms: the C ''\[bfrnt]'' form; the control character '^' form (e.g., ''^@''); or hexadecimal leading ''\x'' form (e.g., ''\xab''). Space is non-printable in the COMMAND column (''\x20'') and printable elsewhere.

For some dialects - if HASSETLOCALE is defined in the dialect's machine.h header file - *lsof* will

print the extended 8 bit characters of a language locale. The *lsof* process must be supplied a language locale environment variable (e.g., LANG) whose value represents a known language locale in which the extended characters are considered printable by *isprint*(3). Otherwise *lsof* considers the extended characters non-printable and prints them according to its rules for non-printable characters, stated above. Consult your dialect's *setlocale*(3) man page for the names of other environment variables that may be used in place of LANG - e.g., LC_ALL, LC_CTYPE, etc.

Lsof's language locale support for a dialect also covers wide characters - e.g., UTF-8 - when HASSETLOCALE and HASWIDECHAR are defined in the dialect's machine.h header file, and when a suitable language locale has been defined in the appropriate environment variable for the *lsof* process. Wide characters are printable under those conditions if *iswprint*(3) reports them to be. If HASSETLOCALE, HASWIDECHAR and a suitable language locale aren't defined, or if *iswprint*(3) reports wide characters that aren't printable, *lsof* considers the wide characters non-printable and prints each of their 8 bits according to its rules for non-printable characters, stated above.

Consult the answers to the "Language locale support" questions in the lsof FAQ (The **FAQ** section gives its location.) for more information.

Lsof dynamically sizes the output columns each time it runs, guaranteeing that each column is a minimum size. It also guarantees that each column is separated from its predecessor by at least one space.

COMMAND

contains the first nine characters of the name of the UNIX command associated with the process. If a non-zero *w* value is specified to the **+c** *w* option, the column contains the first *w* characters of the name of the UNIX command associated with the process up to the limit of characters supplied to *lsof* by the UNIX dialect. (See the description of the **+c** *w* command or the *lsof* FAQ for more information. The **FAQ** section gives its location.)

If *w* is less than the length of the column title, ''COMMAND'', it will be raised to that length.

If a zero *w* value is specified to the **+c** *w* option, the column contains all the characters of the name of the UNIX command associated with the process.

All command name characters maintained by the kernel in its structures are displayed in field output when the command name descriptor ('c') is specified. See the **OUTPUT FOR OTHER COMMANDS** section for information on selecting field output and the associated command name descriptor.

PID is the Process IDentification number of the process.

TID is the task (thread) IDentification number, if task (thread) reporting is supported by the dialect and a task (thread) is being listed. (If help output - i.e., the output of the **-h** or **-?** options - shows this option, then task (thread) reporting is supported by the dialect.)

A blank TID column in Linux indicates a process - i.e., a non-task.

TASKCMD
is the task command name. Generally this will be the same as the process named in the COMMAND column, but some task implementations (e.g., Linux) permit a task to change its command name.

The TASKCMD column width is subject to the same size limitation as the COMMAND column.

ZONE is the Solaris 10 and higher zone name. This column must be selected with the **-z** option.

SECURITY-CONTEXT
is the SELinux security context. This column must be selected with the **-Z** option. Note that the **-Z** option is inhibited when SELinux is disabled in the running Linux kernel.

PPID is the Parent Process IDentification number of the process. It is only displayed when the **-R** option has been specified.

PGID is the process group IDentification number associated with the process. It is only displayed when the **-g** option has been specified.

USER is the user ID number or login name of the user to whom the process belongs, usually the same as reported by *ps*(1). However, on Linux USER is the user ID number or login that owns the directory in /proc where *lsof* finds information about the process. Usually that is the same value reported by *ps*(1), but may differ when the process has changed its effective user ID. (See the **-l** option description for information on when a user ID number or login name is displayed.)

FD is the File Descriptor number of the file or:

cwd current working directory;
L*nn* library references (AIX);
err FD information error (see NAME column);

jld jail directory (FreeBSD);
ltx shared library text (code and data);
Mxx hex memory-mapped type number xx.
m86 DOS Merge mapped file;
mem memory-mapped file;
mmap memory-mapped device;
pd parent directory;
rtd root directory;
tr kernel trace file (OpenBSD);
txt program text (code and data);
v86 VP/ix mapped file;

FD is followed by one of these characters, describing the mode under which the file is open:

r for read access;
w for write access;
u for read and write access;
space if mode unknown and no lock character follows;
'-' if mode unknown and lock character follows.

The mode character is followed by one of these lock characters, describing the type of lock applied to the file:

N for a Solaris NFS lock of unknown type;
r for read lock on part of the file;
R for a read lock on the entire file;
w for a write lock on part of the file;
W for a write lock on the entire file;
u for a read and write lock of any length;
U for a lock of unknown type;
x for an SCO OpenServer Xenix lock on part of the file;
X for an SCO OpenServer Xenix lock on the entire file;
space if there is no lock.

See the **LOCKS** section for more information on the lock information character.

The FD column contents constitutes a single field for parsing in post-processing scripts. FD numbers larger than 9999 are abbreviated to a ''*'' followed by the last three digits. E.g.,

10001 appears as ‘‘*001’’

TYPE is the type of the node associated with the file - e.g., GDIR, GREG, VDIR, VREG, etc.

or ‘‘IPv4’’ for an IPv4 socket;

or ‘‘IPv6’’ for an open IPv6 network file - even if its address is IPv4, mapped in an IPv6 address;

or ‘‘ax25’’ for a Linux AX.25 socket;

or ‘‘inet’’ for an Internet domain socket;

or ‘‘lla’’ for a HP-UX link level access file;

or ‘‘rte’’ for an AF_ROUTE socket;

or ‘‘sock’’ for a socket of unknown domain;

or ‘‘unix’’ for a UNIX domain socket;

or ‘‘x.25’’ for an HP-UX x.25 socket;

or ‘‘BLK’’ for a block special file;

or ‘‘CHR’’ for a character special file;

or ‘‘DEL’’ for a Linux map file that has been deleted;

or ‘‘DIR’’ for a directory;

or ‘‘DOOR’’ for a VDOOR file;

or ‘‘FIFO’’ for a FIFO special file;

or ‘‘KQUEUE’’ for a BSD style kernel event queue file;

or ‘‘LINK’’ for a symbolic link file;

or ‘‘MPB’’ for a multiplexed block file;

or ‘‘MPC’’ for a multiplexed character file;

or ‘‘NOFD’’ for a Linux /proc/<PID>/fd directory that can’t be opened -- the directory path appears in the NAME column, followed by an error message;

or ‘‘PAS’’ for a */proc/as* file;

or ‘‘PAXV’’ for a */proc/auxv* file;

or ‘‘PCRE’’ for a */proc/cred* file;

or ‘‘PCTL’’ for a */proc* control file;

or ‘‘PCUR’’ for the current */proc* process;

or ‘‘PCWD’’ for a */proc* current working directory;

or ‘‘PDIR’’ for a */proc* directory;

or ‘‘PETY’’ for a */proc* executable type (*etype*);

or ‘‘PFD’’ for a */proc* file descriptor;

or ‘‘PFDR’’ for a */proc* file descriptor directory;

or ‘‘PFIL’’ for an executable */proc* file;

or ‘‘PFPR’’ for a */proc* FP register set;

or ‘‘PGD’’ for a */proc/pagedata* file;

or ‘‘PGID’’ for a */proc* group notifier file;

or ‘‘PIPE’’ for pipes;

or ‘‘PLC’’ for a */proc/lwpctl* file;

or ‘‘PLDR’’ for a */proc/lpw* directory;

or ‘‘PLDT’’ for a */proc/ldt* file;

or ‘‘PLPI’’ for a */proc/lpsinfo* file;

or ‘‘PLST’’ for a */proc/lstatus* file;

or ‘‘PLU’’ for a */proc/lusage* file;

or ‘‘PLWG’’ for a */proc/gwindows* file;

or ‘‘PLWI’’ for a */proc/lwpsinfo* file;

or ‘‘PLWS’’ for a */proc/lwpstatus* file;

or ‘‘PLWU’’ for a */proc/lwpusage* file;

or ‘‘PLWX’’ for a */proc/xregs* file;

or ‘‘PMAP’’ for a */proc* map file (*map*);

or ‘‘PMEM’’ for a */proc* memory image file;

or ‘‘PNTF’’ for a */proc* process notifier file;

or ‘‘POBJ’’ for a */proc/object* file;

or ‘‘PODR’’ for a */proc/object* directory;

or ‘‘POLP’’ for an old format */proc* light weight process file;

or ‘‘POPF’’ for an old format */proc* PID file;

or ‘‘POPG’’ for an old format */proc* page data file;

or ‘‘PORT’’ for a SYSV named pipe;

or ‘‘PREG’’ for a */proc* register file;

or ‘‘PRMP’’ for a */proc/rmap* file;

or ‘‘PRTD’’ for a */proc* root directory;

or ‘‘PSGA’’ for a */proc/sigact* file;

or ‘‘PSIN’’ for a */proc/psinfo* file;

or ‘‘PSTA’’ for a */proc* status file;

or ‘‘PSXMQ’’ for a POSIX message queue file;

or ‘‘PSXSEM’’ for a POSIX semaphore file;

or ‘‘PSXSHM’’ for a POSIX shared memory file;

or ‘‘PTS’’ for a */dev/pts* file;

or ‘‘PUSG’’ for a */proc/usage* file;

or ‘‘PW’’ for a */proc/watch* file;

or ‘‘PXMP’’ for a */proc/xmap* file;

or ‘‘REG’’ for a regular file;

or ‘‘SMT’’ for a shared memory transport file;

or ‘‘STSO’’ for a stream socket;

or ‘‘UNNM’’ for an unnamed type file;

or ‘‘XNAM’’ for an OpenServer Xenix special file of unknown type;

or ‘‘XSEM’’ for an OpenServer Xenix semaphore file;

or ‘‘XSD’’ for an OpenServer Xenix shared data file;

or the four type number octets if the corresponding name isn't known.

FILE-ADDR
contains the kernel file structure address when **f** has been specified to **+f**;

FCT contains the file reference count from the kernel file structure when **c** has been specified to

+f;

FILE-FLAG

when **g** or **G** has been specified to **+f**, this field contains the contents of the f_flag[s] member of the kernel file structure and the kernel's per-process open file flags (if available); 'G' causes them to be displayed in hexadecimal; 'g', as short-hand names; two lists may be displayed with entries separated by commas, the lists separated by a semicolon (';'); the first list may contain short-hand names for f_flag[s] values from the following table:

AIO	asynchronous I/O (e.g., FAIO)
AP	append
ASYN	asynchronous I/O (e.g., FASYNC)
BAS	block, test, and set in use
BKIU	block if in use
BL	use block offsets
BSK	block seek
CA	copy avoid
CIO	concurrent I/O
CLON	clone
CLRD	CL read
CR	create
DF	defer
DFI	defer IND
DFLU	data flush
DIR	direct
DLY	delay
DOCL	do clone
DSYN	data-only integrity
DTY	must be a directory
EVO	event only
EX	open for exec
EXCL	exclusive open
FSYN	synchronous writes
GCDF	defer during unp_gc() (AIX)
GCMK	mark during unp_gc() (AIX)
GTTY	accessed via /dev/tty
HUP	HUP in progress
KERN	kernel
KIOC	kernel-issued ioctl

LCK	has lock
LG	large file
MBLK	stream message block
MK	mark
MNT	mount
MSYN	multiplex synchronization
NATM	don't update atime
NB	non-blocking I/O
NBDR	no BDRM check
NBIO	SYSV non-blocking I/O
NBF	n-buffering in effect
NC	no cache
ND	no delay
NDSY	no data synchronization
NET	network
NFLK	don't follow links
NMFS	NM file system
NOTO	disable background stop
NSH	no share
NTTY	no controlling TTY
OLRM	OLR mirror
PAIO	POSIX asynchronous I/O
PATH	path
PP	POSIX pipe
R	read
RC	file and record locking cache
REV	revoked
RSH	shared read
RSYN	read synchronization
RW	read and write access
SL	shared lock
SNAP	cooked snapshot
SOCK	socket
SQSH	Sequent shared set on open
SQSV	Sequent SVM set on open
SQR	Sequent set repair on open
SQS1	Sequent full shared open
SQS2	Sequent partial shared open
STPI	stop I/O
SWR	synchronous read

SYN	file integrity while writing
TCPM	avoid TCP collision
TMPF	temporary file
TR	truncate
W	write
WKUP	parallel I/O synchronization
WTG	parallel I/O synchronization
VH	vhangup pending
VTXT	virtual text
XL	exclusive lock

this list of names was derived from F* #define's in dialect header files <fcntl.h>, <linux</fs.h>, <sys/fcntl.c>, <sys/fcntlcom.h>, and <sys/file.h>; see the lsof.h header file for a list showing the correspondence between the above short-hand names and the header file definitions;

the second list (after the semicolon) may contain short-hand names for kernel per-process open file flags from this table:

ALLC	allocated
BR	the file has been read
BHUP	activity stopped by SIGHUP
BW	the file has been written
CLSG	closing
CX	close-on-exec (see fcntl(F_SETFD))
LCK	lock was applied
MP	memory-mapped
OPIP	open pending - in progress
RSVW	reserved wait
SHMT	UF_FSHMAT set (AIX)
USE	in use (multi-threaded)

NODE-ID (or INODE-ADDR for some dialects) contains a unique identifier for the file node (usually the kernel vnode or inode address, but also occasionally a concatenation of device and node number) when **n** has been specified to **+f**;

DEVICE contains the device numbers, separated by commas, for a character special, block special, regular, directory or NFS file;

or ''memory'' for a memory file system node under Tru64 UNIX;

or the address of the private data area of a Solaris socket stream;

or a kernel reference address that identifies the file (The kernel reference address may be used for FIFO's, for example.);

or the base address or device name of a Linux AX.25 socket device.

Usually only the lower thirty two bits of Tru64 UNIX kernel addresses are displayed.

SIZE, SIZE/OFF, or OFFSET

is the size of the file or the file offset in bytes. A value is displayed in this column only if it is available. *Lsof* displays whatever value - size or offset - is appropriate for the type of the file and the version of *lsof*.

On some UNIX dialects *lsof* can't obtain accurate or consistent file offset information from its kernel data sources, sometimes just for particular kinds of files (e.g., socket files.) In other cases, files don't have true sizes - e.g., sockets, FIFOs, pipes - so *lsof* displays for their sizes the content amounts it finds in their kernel buffer descriptors (e.g., socket buffer size counts or TCP/IP window sizes.) Consult the *lsof* FAQ (The **FAQ** section gives its location.) for more information.

The file size is displayed in decimal; the offset is normally displayed in decimal with a leading ''0t'' if it contains 8 digits or less; in hexadecimal with a leading ''0x'' if it is longer than 8 digits. (Consult the **-o** *o* option description for information on when 8 might default to some other value.)

Thus the leading ''0t'' and ''0x'' identify an offset when the column may contain both a size and an offset (i.e., its title is SIZE/OFF).

If the **-o** option is specified, *lsof* always displays the file offset (or nothing if no offset is available) and labels the column OFFSET. The offset always begins with ''0t'' or ''0x'' as described above.

The *lsof* user can control the switch from ''0t'' to ''0x'' with the **-o** *o* option. Consult its description for more information.

If the **-s** option is specified, *lsof* always displays the file size (or nothing if no size is available) and labels the column SIZE. The **-o** and **-s** options are mutually exclusive; they can't both be specified.

For files that don't have a fixed size - e.g., don't reside on a disk device - *lsof* will display appropriate information about the current size or position of the file if it is available in the kernel structures that define the file.

NLINK contains the file link count when **+L** has been specified;

NODE is the node number of a local file;

or the inode number of an NFS file in the server host;

or the Internet protocol type - e.g, ''TCP'';

or ''STR'' for a stream;

or ''CCITT'' for an HP-UX x.25 socket;

or the IRQ or inode number of a Linux AX.25 socket device.

NAME is the name of the mount point and file system on which the file resides;

or the name of a file specified in the *names* option (after any symbolic links have been resolved);

or the name of a character special or block special device;

or the local and remote Internet addresses of a network file; the local host name or IP number is followed by a colon (':'), the port, ''->'', and the two-part remote address; IP addresses may be reported as numbers or names, depending on the **+|-M**, **-n**, and **-P** options; colon-separated IPv6 numbers are enclosed in square brackets; IPv4 INADDR_ANY and IPv6 IN6_IS_ADDR_UNSPECIFIED addresses, and zero port numbers are represented by an asterisk ('*'); a UDP destination address may be followed by the amount of time elapsed since the last packet was sent to the destination; TCP, UDP and UDPLITE remote addresses may be followed by TCP/TPI information in parentheses - state (e.g., ''(ESTABLISHED)'', ''(Unbound)''), queue sizes, and window sizes (not all dialects) - in a fashion similar to what *netstat*(1) reports; see the **-T** option description or the description of the TCP/TPI field in **OUTPUT FOR OTHER PROGRAMS** for more information on state, queue size, and window size;

or the address or name of a UNIX domain socket, possibly including a stream clone device name, a file system object's path name, local and foreign kernel addresses, socket pair

information, and a bound vnode address;

or the local and remote mount point names of an NFS file;

or ''STR'', followed by the stream name;

or a stream character device name, followed by ''->'' and the stream name or a list of stream module names, separated by ''->'';

or ''STR:'' followed by the SCO OpenServer stream device and module names, separated by ''->'';

or system directory name, '' -- '', and as many components of the path name as *lsof* can find in the kernel's name cache for selected dialects (See the **KERNEL NAME CACHE** section for more information.);

or ''PIPE->'', followed by a Solaris kernel pipe destination address;

or ''COMMON:'', followed by the vnode device information structure's device name, for a Solaris common vnode;

or the address family, followed by a slash ('/'), followed by fourteen comma-separated bytes of a non-Internet raw socket address;

or the HP-UX x.25 local address, followed by the virtual connection number (if any), followed by the remote address (if any);

or ''(dead)'' for disassociated Tru64 UNIX files - typically terminal files that have been flagged with the TIOCNOTTY ioctl and closed by daemons;

or ''rd=<offset>'' and ''wr=<offset>'' for the values of the read and write offsets of a FIFO;

or ''clone *n*:/dev/event'' for SCO OpenServer file clones of the */dev/event* device, where *n* is the minor device number of the file;

or ''(socketpair: n)'' for a Solaris 2.6, 8, 9 or 10 UNIX domain socket, created by the *socketpair*(3N) network function;

or ''no PCB'' for socket files that do not have a protocol block associated with them,

optionally followed by ‘‘, CANTSENDMORE’’ if sending on the socket has been disabled, or ‘‘, CANTRCVMORE’’ if receiving on the socket has been disabled (e.g., by the *shutdown*(2) function);

or the local and remote addresses of a Linux IPX socket file in the form <net>:[<node>:]<port>, followed in parentheses by the transmit and receive queue sizes, and the connection state;

or ‘‘dgram’’ or ‘‘stream’’ for the type UnixWare 7.1.1 and above in-kernel UNIX domain sockets, followed by a colon (’:’) and the local path name when available, followed by ‘‘->’’ and the remote path name or kernel socket address in hexadecimal when available;

or the association value, association index, endpoint value, local address, local port, remote address and remote port for Linux SCTP sockets;

or ‘‘protocol: ’’ followed by the Linux socket’s protocol attribute.

For dialects that support a ‘‘namefs’’ file system, allowing one file to be attached to another with *fattach*(3C), *lsof* will add ‘‘(FA:<address1><direction><address2>)’’ to the NAME column. <address1> and <address2> are hexadecimal vnode addresses. <direction> will be ‘‘<-’’ if <address2> has been fattach’ed to this vnode whose address is <address1>; and ‘‘->’’ if <address1>, the vnode address of this vnode, has been fattach’ed to <address2>. <address1> may be omitted if it already appears in the DEVICE column.

Lsof may add two parenthetical notes to the NAME column for open Solaris 10 files: ‘‘(?)’’ if *lsof* considers the path name of questionable accuracy; and ‘‘(deleted)’’ if the **-X** option has been specified and *lsof* detects the open file’s path name has been deleted. Consult the *lsof* FAQ (The **FAQ** section gives its location.) for more information on these NAME column additions.

LOCKS

Lsof can’t adequately report the wide variety of UNIX dialect file locks in a single character. What it reports in a single character is a compromise between the information it finds in the kernel and the limitations of the reporting format.

Moreover, when a process holds several byte level locks on a file, *lsof* only reports the status of the first lock it encounters. If it is a byte level lock, then the lock character will be reported in lower case - i.e., ‘r’, ‘w’, or ‘x’ - rather than the upper case equivalent reported for a full file lock.

Generally *lsof* can only report on locks held by local processes on local files. When a local process sets a lock on a remotely mounted (e.g., NFS) file, the remote server host usually records the lock state.

One exception is Solaris - at some patch levels of 2.3, and in all versions above 2.4, the Solaris kernel records information on remote locks in local structures.

Lsof has trouble reporting locks for some UNIX dialects. Consult the **BUGS** section of this manual page or the *lsof* FAQ (The **FAQ** section gives its location.) for more information.

OUTPUT FOR OTHER PROGRAMS

When the **-F** option is specified, *lsof* produces output that is suitable for processing by another program - e.g, an *awk* or *Perl* script, or a C program.

Each unit of information is output in a field that is identified with a leading character and terminated by a NL (012) (or a NUL (000) if the 0 (zero) field identifier character is specified.) The data of the field follows immediately after the field identification character and extends to the field terminator.

It is possible to think of field output as process and file sets. A process set begins with a field whose identifier is 'p' (for process IDentifier (PID)). It extends to the beginning of the next PID field or the beginning of the first file set of the process, whichever comes first. Included in the process set are fields that identify the command, the process group IDentification (PGID) number, the task (thread) ID (TID), and the user ID (UID) number or login name.

A file set begins with a field whose identifier is 'f' (for file descriptor). It is followed by lines that describe the file's access mode, lock state, type, device, size, offset, inode, protocol, name and stream module names. It extends to the beginning of the next file or process set, whichever comes first.

When the NUL (000) field terminator has been selected with the 0 (zero) field identifier character, *lsof* ends each process and file set with a NL (012) character.

Lsof always produces one field, the PID ('p') field. In repeat mode, the marker ('m') is also produced. All other fields may be declared optionally in the field identifier character list that follows the **-F** option. When a field selection character identifies an item *lsof* does not normally list - e.g., PPID, selected with **-R** - specification of the field character - e.g., ''**-FR**'' - also selects the listing of the item.

Lsof version from 4.88 to 4.93.2 always produced one more field, the file descriptor ('f') field. However, *lsof* in this version doesn't produce it. This change is for supporting the use case that a user needs only the PID field, and doesn't need the file descriptor field. Specify 'f' explicitly if you need the field.

It is entirely possible to select a set of fields that cannot easily be parsed - e.g., if the field descriptor field is not selected, it may be difficult to identify file sets. To help you avoid this difficulty, *lsof* supports the **-F** option; it selects the output of all fields with NL terminators (the **-F0** option pair selects

the output of all fields with NUL terminators). For compatibility reasons neither **-F** nor **-F0** select the raw device field.

These are the fields that *lsof* will produce. The single character listed first is the field identifier.

a file access mode
c process command name (all characters from proc or user structure)
C file structure share count
d file's device character code
D file's major/minor device number (0x<hexadecimal>)
f file descriptor
F file structure address (0x<hexadecimal>)
G file flaGs (0x<hexadecimal>; names if **+fg** follows)
g process group ID
i file's inode number
K tasK ID
k link count
l file's lock status
L process login name
m marker between repeated output (always selected in repeat mode)
M the task comMand name
n file name, comment, Internet address
N node identifier (ox<hexadecimal>
o file's offset (0t<decimal> or 0x<hexadecimal>, see **-o** *o*)
p process ID (always selected)
P protocol name
r raw device number (0x<hexadecimal>)
R parent process ID
s file's size (decimal)
S file's stream identification
t file's type
T TCP/TPI information, identified by prefixes (the '=' is part of the prefix):
QR=<read queue size>
QS=<send queue size>
SO=<socket options and values> (not all dialects)
SS=<socket states> (not all dialects)
ST=<connection state>
TF=<TCP flags and values> (not all dialects)

WR=<window read size> (not all dialects)
WW=<window write size> (not all dialects)
(TCP/TPI information isn't reported for all supported UNIX dialects. The **-h** or **-?** help output for the **-T** option will show what TCP/TPI reporting can be requested.)

u process user ID
z Solaris 10 and higher zone name
Z SELinux security context (inhibited when SELinux is disabled)
0 use NUL field terminator character in place of NL
1-9 dialect-specific field identifiers (The output of **-F?** identifies the information to be found in dialect-specific fields.)

You can get on-line help information on these characters and their descriptions by specifying the **-F?** option pair. (Escape the '?' character as your shell requires.) Additional information on field content can be found in the **OUTPUT** section.

As an example, ''**-F pcfn**'' will select the process ID ('p'), command name ('c'), file descriptor ('f') and file name ('n') fields with an NL field terminator character; ''**-F pcfn0**'' selects the same output with a NUL (000) field terminator character.

Lsof doesn't produce all fields for every process or file set, only those that are available. Some fields are mutually exclusive: file device characters and file major/minor device numbers; file inode number and protocol name; file name and stream identification; file size and offset. One or the other member of these mutually exclusive sets will appear in field output, but not both.

Normally *lsof* ends each field with a NL (012) character. The 0 (zero) field identifier character may be specified to change the field terminator character to a NUL (000). A NUL terminator may be easier to process with *xargs (1)*, for example, or with programs whose quoting mechanisms may not easily cope with the range of characters in the field output. When the NUL field terminator is in use, *lsof* ends each process and file set with a NL (012).

Three aids to producing programs that can process *lsof* field output are included in the *lsof* distribution. The first is a C header file, *lsof_fields.h*, that contains symbols for the field identification characters, indexes for storing them in a table, and explanation strings that may be compiled into programs. *Lsof* uses this header file.

The second aid is a set of sample scripts that process field output, written in *awk*, *Perl* 4, and *Perl* 5. They're located in the *scripts* subdirectory of the *lsof* distribution.

The third aid is the C library used for the *lsof* test suite. The test suite is written in C and uses field output to validate the correct operation of *lsof*. The library can be found in the *tests/LTlib.c* file of the *lsof* distribution. The library uses the first aid, the *lsof_fields.h* header file.

BLOCKS AND TIMEOUTS

Lsof can be blocked by some kernel functions that it uses - *lstat*(2), *readlink*(2), and *stat*(2). These functions are stalled in the kernel, for example, when the hosts where mounted NFS file systems reside become inaccessible.

Lsof attempts to break these blocks with timers and child processes, but the techniques are not wholly reliable. When *lsof* does manage to break a block, it will report the break with an error message. The messages may be suppressed with the **-t** and **-w** options.

The default timeout value may be displayed with the **-h** or **-?** option, and it may be changed with the **-S** *[t]* option. The minimum for *t* is two seconds, but you should avoid small values, since slow system responsiveness can cause short timeouts to expire unexpectedly and perhaps stop *lsof* before it can produce any output.

When *lsof* has to break a block during its access of mounted file system information, it normally continues, although with less information available to display about open files.

Lsof can also be directed to avoid the protection of timers and child processes when using the kernel functions that might block by specifying the **-O** option. While this will allow *lsof* to start up with less overhead, it exposes *lsof* completely to the kernel situations that might block it. Use this option cautiously.

AVOIDING KERNEL BLOCKS

You can use the **-b** option to tell *lsof* to avoid using kernel functions that would block. Some cautions apply.

First, using this option usually requires that your system supply alternate device numbers in place of the device numbers that *lsof* would normally obtain with the *lstat*(2) and *stat*(2) kernel functions. See the **ALTERNATE DEVICE NUMBERS** section for more information on alternate device numbers.

Second, you can't specify *names* for *lsof* to locate unless they're file system names. This is because *lsof* needs to know the device and inode numbers of files listed with *names* in the *lsof* options, and the **-b** option prevents *lsof* from obtaining them. Moreover, since *lsof* only has device numbers for the file systems that have alternates, its ability to locate files on file systems depends completely on the availability and accuracy of the alternates. If no alternates are available, or if they're incorrect, *lsof* won't be able to locate files on the named file systems.

Third, if the names of your file system directories that *lsof* obtains from your system's mount table are symbolic links, *lsof* won't be able to resolve the links. This is because the **-b** option causes *lsof* to avoid the kernel *readlink*(2) function it uses to resolve symbolic links.

Finally, using the **-b** option causes *lsof* to issue warning messages when it needs to use the kernel functions that the **-b** option directs it to avoid. You can suppress these messages by specifying the **-w** option, but if you do, you won't see the alternate device numbers reported in the warning messages.

ALTERNATE DEVICE NUMBERS

On some dialects, when *lsof* has to break a block because it can't get information about a mounted file system via the *lstat*(2) and *stat*(2) kernel functions, or because you specified the **-b** option, *lsof* can obtain some of the information it needs - the device number and possibly the file system type - from the system mount table. When that is possible, *lsof* will report the device number it obtained. (You can suppress the report by specifying the **-w** option.)

You can assist this process if your mount table is supported with an */etc/mtab* or */etc/mnttab* file that contains an options field by adding a ''dev=xxxx'' field for mount points that do not have one in their options strings. Note: you must be able to edit the file - i.e., some mount tables like recent Solaris /etc/mnttab or Linux /proc/mounts are read-only and can't be modified.

You may also be able to supply device numbers using the **+m** and **+m** *m* options, provided they are supported by your dialect. Check the output of *lsof's* **-h** or **-?** options to see if the **+m** and **+m** *m* options are available.

The ''xxxx'' portion of the field is the hexadecimal value of the file system's device number. (Consult the *st_dev* field of the output of the *lstat*(2) and *stat*(2) functions for the appropriate values for your file systems.) Here's an example from a Sun Solaris 2.6 */etc/mnttab* for a file system remotely mounted via NFS:

```
nfs  ignore,noquota,dev=2a40001
```

There's an advantage to having ''dev=xxxx'' entries in your mount table file, especially for file systems that are mounted from remote NFS servers. When a remote server crashes and you want to identify its users by running *lsof* on one of its clients, *lsof* probably won't be able to get output from the *lstat*(2) and *stat*(2) functions for the file system. If it can obtain the file system's device number from the mount table, it will be able to display the files open on the crashed NFS server.

Some dialects that do not use an ASCII */etc/mtab* or */etc/mnttab* file for the mount table may still provide an alternative device number in their internal mount tables. This includes AIX, Apple Darwin, FreeBSD, NetBSD, OpenBSD, and Tru64 UNIX. *Lsof* knows how to obtain the alternative device

number for these dialects and uses it when its attempt to *lstat*(2) or *stat*(2) the file system is blocked.

If you're not sure your dialect supplies alternate device numbers for file systems from its mount table, use this *lsof* incantation to see if it reports any alternate device numbers:

```
lsof -b
```

Look for standard error file warning messages that begin ''assuming "dev=xxxx" from ...''.

KERNEL NAME CACHE

Lsof is able to examine the kernel's name cache or use other kernel facilities (e.g., the ADVFS 4.x tag_to_path() function under Tru64 UNIX) on some dialects for most file system types, excluding AFS, and extract recently used path name components from it. (AFS file system path lookups don't use the kernel's name cache; some Solaris VxFS file system operations apparently don't use it, either.)

Lsof reports the complete paths it finds in the NAME column. If *lsof* can't report all components in a path, it reports in the NAME column the file system name, followed by a space, two '-' characters, another space, and the name components it has located, separated by the '/' character.

When *lsof* is run in repeat mode - i.e., with the **-r** option specified - the extent to which it can report path name components for the same file may vary from cycle to cycle. That's because other running processes can cause the kernel to remove entries from its name cache and replace them with others.

Lsof's use of the kernel name cache to identify the paths of files can lead it to report incorrect components under some circumstances. This can happen when the kernel name cache uses device and node number as a key (e.g., SCO OpenServer) and a key on a rapidly changing file system is reused. If the UNIX dialect's kernel doesn't purge the name cache entry for a file when it is unlinked, *lsof* may find a reference to the wrong entry in the cache. The *lsof* FAQ (The **FAQ** section gives its location.) has more information on this situation.

Lsof can report path name components for these dialects:

FreeBSD
HP-UX
Linux
NetBSD
OpenBSD
SCO OpenServer
SCO|Caldera UnixWare
Solaris

Tru64 UNIX

Lsof can't report path name components for these dialects:

AIX

If you want to know why *lsof* can't report path name components for some dialects, see the *lsof* FAQ (The **FAQ** section gives its location.)

DEVICE CACHE FILE

Examining all members of the */dev* (or */devices*) node tree with *stat*(2) functions can be time consuming. What's more, the information that *lsof* needs - device number, inode number, and path - rarely changes.

Consequently, *lsof* normally maintains an ASCII text file of cached */dev* (or */devices*) information (exception: the /proc-based Linux *lsof* where it's not needed.) The local system administrator who builds *lsof* can control the way the device cache file path is formed, selecting from these options:

Path from the **-D** option;
Path from an environment variable;
System-wide path;
Personal path (the default);
Personal path, modified by an environment variable.

Consult the output of the **-h**, **-D? ,** or **-?** help options for the current state of device cache support. The help output lists the default read-mode device cache file path that is in effect for the current invocation of *lsof*. The **-D?** option output lists the read-only and write device cache file paths, the names of any applicable environment variables, and the personal device cache path format.

Lsof can detect that the current device cache file has been accidentally or maliciously modified by integrity checks, including the computation and verification of a sixteen bit Cyclic Redundancy Check (CRC) sum on the file's contents. When *lsof* senses something wrong with the file, it issues a warning and attempts to remove the current cache file and create a new copy, but only to a path that the process can legitimately write.

The path from which a *lsof* process may attempt to read a device cache file may not be the same as the path to which it can legitimately write. Thus when *lsof* senses that it needs to update the device cache file, it may choose a different path for writing it from the path from which it read an incorrect or outdated version.

If available, the **-Dr** option will inhibit the writing of a new device cache file. (It's always available when specified without a path name argument.)

When a new device is added to the system, the device cache file may need to be recreated. Since *lsof* compares the mtime of the device cache file with the mtime and ctime of the */dev* (or */devices*) directory, it usually detects that a new device has been added; in that case *lsof* issues a warning message and attempts to rebuild the device cache file.

Whenever *lsof* writes a device cache file, it sets its ownership to the real UID of the executing process, and its permission modes to 0600, this restricting its reading and writing to the file's owner.

LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS

Two permissions of the *lsof* executable affect its ability to access device cache files. The permissions are set by the local system administrator when *lsof* is installed.

The first and rarer permission is setuid-root. It comes into effect when *lsof* is executed; its effective UID is then root, while its real (i.e., that of the logged-on user) UID is not. The *lsof* distribution recommends that versions for these dialects run setuid-root.

HP-UX 11.11 and 11.23
Linux

The second and more common permission is setgid. It comes into effect when the effective group IDentification number (GID) of the *lsof* process is set to one that can access kernel memory devices - e.g., ''kmem'', ''sys'', or ''system''.

An *lsof* process that has setgid permission usually surrenders the permission after it has accessed the kernel memory devices. When it does that, *lsof* can allow more liberal device cache path formations. The *lsof* distribution recommends that versions for these dialects run setgid and be allowed to surrender setgid permission.

AIX 5.[12] and 5.3-ML1
Apple Darwin 7.x Power Macintosh systems
FreeBSD 4.x, 4.1x, 5.x and [6789].x for x86-based systems
FreeBSD 5.x, [6789].x and 1[012].8for Alpha, AMD64 and Sparc64 based systems
HP-UX 11.00
NetBSD 1.[456], 2.x and 3.x for Alpha, x86, and SPARC-based systems
OpenBSD 2.[89] and 3.[0-9] for x86-based systems

SCO OpenServer Release 5.0.6 for x86-based systems
SCO|Caldera UnixWare 7.1.4 for x86-based systems
Solaris 2.6, 8, 9 and 10
Tru64 UNIX 5.1

(Note: *lsof* for AIX 5L and above needs setuid-root permission if its **-X** option is used.)

Lsof for these dialects does not support a device cache, so the permissions given to the executable don't apply to the device cache file.

Linux

DEVICE CACHE FILE PATH FROM THE -D OPTION

The **-D** option provides limited means for specifying the device cache file path. Its **?** function will report the read-only and write device cache file paths that *lsof* will use.

When the **-D b**, **r**, and **u** functions are available, you can use them to request that the cache file be built in a specific location (**b**[*path*]); read but not rebuilt (**r**[*path*]); or read and rebuilt (**u**[*path*]). The **b**, **r**, and **u** functions are restricted under some conditions. They are restricted when the *lsof* process is setuid-root. The path specified with the **r** function is always read-only, even when it is available.

The **b**, **r**, and **u** functions are also restricted when the *lsof* process runs setgid and *lsof* doesn't surrender the setgid permission. (See the **LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS** section for a list of implementations that normally don't surrender their setgid permission.)

A further **-D** function, **i** (for ignore), is always available.

When available, the **b** function tells *lsof* to read device information from the kernel with the *stat*(2) function and build a device cache file at the indicated path.

When available, the **r** function tells *lsof* to read the device cache file, but not update it. When a path argument accompanies **-Dr**, it names the device cache file path. The **r** function is always available when it is specified without a path name argument. If *lsof* is not running setuid-root and surrenders its setgid permission, a path name argument may accompany the **r** function.

When available, the **u** function tells *lsof* to attempt to read and use the device cache file. If it can't read the file, or if it finds the contents of the file incorrect or outdated, it will read information from the kernel, and attempt to write an updated version of the device cache file, but only to a path it considers legitimate for the *lsof* process effective and real UIDs.

DEVICE CACHE PATH FROM AN ENVIRONMENT VARIABLE

Lsof's second choice for the device cache file is the contents of the LSOFDEVCACHE environment variable. It avoids this choice if the *lsof* process is setuid-root, or the real UID of the process is root.

A further restriction applies to a device cache file path taken from the LSOFDEVCACHE environment variable: *lsof* will not write a device cache file to the path if the *lsof* process doesn't surrender its setgid permission. (See the **LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS** section for information on implementations that don't surrender their setgid permission.)

The local system administrator can disable the use of the LSOFDEVCACHE environment variable or change its name when building *lsof*. Consult the output of **-D?** for the environment variable's name.

SYSTEM-WIDE DEVICE CACHE PATH

The local system administrator may choose to have a system-wide device cache file when building *lsof*. That file will generally be constructed by a special system administration procedure when the system is booted or when the contents of */dev* or */devices*) changes. If defined, it is *lsof's* third device cache file path choice.

You can tell that a system-wide device cache file is in effect for your local installation by examining the *lsof* help option output - i.e., the output from the **-h** or **-?** option.

Lsof will never write to the system-wide device cache file path by default. It must be explicitly named with a **-D** function in a root-owned procedure. Once the file has been written, the procedure must change its permission modes to 0644 (owner-read and owner-write, group-read, and other-read).

PERSONAL DEVICE CACHE PATH (DEFAULT)

The default device cache file path of the *lsof* distribution is one recorded in the home directory of the real UID that executes *lsof*. Added to the home directory is a second path component of the form *.lsof_hostname*.

This is *lsof's* fourth device cache file path choice, and is usually the default. If a system-wide device cache file path was defined when *lsof* was built, this fourth choice will be applied when *lsof* can't find the system-wide device cache file. This is the **only** time *lsof* uses two paths when reading the device cache file.

The *hostname* part of the second component is the base name of the executing host, as returned by *gethostname*(2). The base name is defined to be the characters preceding the first '.' in the *gethostname*(2) output, or all the *gethostname*(2) output if it contains no '.'.

The device cache file belongs to the user ID and is readable and writable by the user ID alone - i.e., its

modes are 0600. Each distinct real user ID on a given host that executes *lsof* has a distinct device cache file. The *hostname* part of the path distinguishes device cache files in an NFS-mounted home directory into which device cache files are written from several different hosts.

The personal device cache file path formed by this method represents a device cache file that *lsof* will attempt to read, and will attempt to write should it not exist or should its contents be incorrect or outdated.

The **-Dr** option without a path name argument will inhibit the writing of a new device cache file.

The **-D?** option will list the format specification for constructing the personal device cache file. The conversions used in the format specification are described in the *00DCACHE* file of the *lsof* distribution.

MODIFIED PERSONAL DEVICE CACHE PATH

If this option is defined by the local system administrator when *lsof* is built, the LSOFPERSDCPATH environment variable contents may be used to add a component of the personal device cache file path.

The LSOFPERSDCPATH variable contents are inserted in the path at the place marked by the local system administrator with the ''%p'' conversion in the HASPERSDC format specification of the dialect's *machine.h* header file. (It's placed right after the home directory in the default *lsof* distribution.)

Thus, for example, if LSOFPERSDCPATH contains ''LSOF'', the home directory is ''/Homes/abe'', the host name is ''lsof.itap.purdue.edu'', and the HASPERSDC format is the default (''%h/%p.lsof_%L''), the modified personal device cache file path is:

/Homes/abe/LSOF/.lsof_vic

The LSOFPERSDCPATH environment variable is ignored when the *lsof* process is setuid-root or when the real UID of the process is root.

Lsof will not write to a modified personal device cache file path if the *lsof* process doesn't surrender setgid permission. (See the **LSOF PERMISSIONS THAT AFFECT DEVICE CACHE FILE ACCESS** section for a list of implementations that normally don't surrender their setgid permission.)

If, for example, you want to create a sub-directory of personal device cache file paths by using the LSOFPERSDCPATH environment variable to name it, and *lsof* doesn't surrender its setgid permission, you will have to allow *lsof* to create device cache files at the standard personal path and move them to your subdirectory with shell commands.

The local system administrator may: disable this option when *lsof* is built; change the name of the environment variable from LSOFPERSDCPATH to something else; change the HASPERSDC format to include the personal path component in another place; or exclude the personal path component entirely. Consult the output of the **-D?** option for the environment variable's name and the HASPERSDC format specification.

DIAGNOSTICS

Errors are identified with messages on the standard error file.

Lsof returns a one (1) if any error was detected, including the failure to locate command names, file names, Internet addresses or files, login names, NFS files, PIDs, PGIDs, or UIDs it was asked to list. If the **-V** option is specified, *lsof* will indicate the search items it failed to list. If the **-Q** option is specified, *lsof* will ignore any search item failures and only return an error if something unusual and unrecoverable happened.

It returns a zero (0) if no errors were detected and if either the **-Q** option was specified or it was able to list some information about all the specified search arguments.

When *lsof* cannot open access to */dev* (or */devices*) or one of its subdirectories, or get information on a file in them with *stat*(2), it issues a warning message and continues. That *lsof* will issue warning messages about inaccessible files in */dev* (or */devices*) is indicated in its help output - requested with the **-h** or >B -? options - with the message:

Inaccessible /dev warnings are enabled.

The warning message may be suppressed with the **-w** option. It may also have been suppressed by the system administrator when *lsof* was compiled by the setting of the WARNDEVACCESS definition. In this case, the output from the help options will include the message:

Inaccessible /dev warnings are disabled.

Inaccessible device warning messages usually disappear after *lsof* has created a working device cache file.

EXAMPLES

For a more extensive set of examples, documented more fully, see the *00QUICKSTART* file of the *lsof* distribution.

To list all open files, use:

lsof

To list all open Internet, x.25 (HP-UX), and UNIX domain files, use:

lsof -i -U

To list all open IPv4 network files in use by the process whose PID is 1234, use:

lsof -i 4 -a -p 1234

If it's okay for PID 1234 to not exist, or for PID 1234 to not have any open IPv4 network files, add **-Q** :

lsof -Q -i 4 -a -p 1234

Presuming the UNIX dialect supports IPv6, to list only open IPv6 network files, use:

lsof -i 6

To list all files using any protocol on ports 513, 514, or 515 of host wonderland.cc.purdue.edu, use:

lsof -i @wonderland.cc.purdue.edu:513-515

To list all files using any protocol on any port of mace.cc.purdue.edu (cc.purdue.edu is the default domain), use:

lsof -i @mace

To list all open files for login name ''abe'', or user ID 1234, or process 456, or process 123, or process 789, use:

lsof -p 456,123,789 -u 1234,abe

To list all open files on device /dev/hd4, use:

lsof /dev/hd4

To find the process that has /u/abe/foo open without worrying if there are none, use:

lsof -Q /u/abe/foo

To take action only if a process has /u/abe/foo open, use:

```
lsof /u/abe/foo  echo "still in use"
```

To send a SIGHUP to the processes that have /u/abe/bar open, use:

```
kill -HUP `lsof -t /u/abe/bar`
```

To find any open file, including an open UNIX domain socket file, with the name */dev/log*, use:

```
lsof /dev/log
```

To find processes with open files on the NFS file system named */nfs/mount/point* whose server is inaccessible, and presuming your mount table supplies the device number for */nfs/mount/point*, use:

```
lsof -b /nfs/mount/point
```

To do the preceding search with warning messages suppressed, use:

```
lsof -bw /nfs/mount/point
```

To ignore the device cache file, use:

```
lsof -Di
```

To obtain PID and command name field output for each process, file descriptor, file device number, and file inode number for each file of each process, use:

```
lsof -FpcfDi
```

To list the files at descriptors 1 and 3 of every process running the *lsof* command for login ID ''abe'' every 10 seconds, use:

```
lsof -c lsof -a -d 1 -d 3 -u abe -r10
```

To list the current working directory of processes running a command that is exactly four characters long and has an 'o' or 'O' in character three, use this regular expression form of the **-c** *c* option:

```
lsof -c /^..o.$/i -a -d cwd
```

To find an IP version 4 socket file by its associated numeric dot-form address, use:

```
lsof -i@128.210.15.17
```

To find an IP version 6 socket file (when the UNIX dialect supports IPv6) by its associated numeric colon-form address, use:

```
lsof -i@[0:1:2:3:4:5:6:7]
```

To find an IP version 6 socket file (when the UNIX dialect supports IPv6) by an associated numeric colon-form address that has a run of zeroes in it - e.g., the loop-back address - use:

```
lsof -i@[::1]
```

To obtain a repeat mode marker line that contains the current time, use:

```
lsof -rm====%T====
```

To add spaces to the previous marker line, use:

```
lsof -r "m==== %T ===="
```

BUGS

Since *lsof* reads kernel memory in its search for open files, rapid changes in kernel memory may produce unpredictable results.

When a file has multiple record locks, the lock status character (following the file descriptor) is derived from a test of the first lock structure, not from any combination of the individual record locks that might be described by multiple lock structures.

Lsof can't search for files with restrictive access permissions by *name* unless it is installed with root set-UID permission. Otherwise it is limited to searching for files to which its user or its set-GID group (if any) has access permission.

The display of the destination address of a raw socket (e.g., for *ping*) depends on the UNIX operating system. Some dialects store the destination address in the raw socket's protocol control block, some do not.

Lsof can't always represent Solaris device numbers in the same way that *ls*(1) does. For example, the major and minor device numbers that the *lstat*(2) and *stat*(2) functions report for the directory on which

CD-ROM files are mounted (typically */cdrom*) are not the same as the ones that it reports for the device on which CD-ROM files are mounted (typically */dev/sr0*). (*Lsof* reports the directory numbers.)

The support for */proc* file systems is available only for BSD and Tru64 UNIX dialects, Linux, and dialects derived from SYSV R4 - e.g., FreeBSD, NetBSD, OpenBSD, Solaris, UnixWare.

Some */proc* file items - device number, inode number, and file size - are unavailable in some dialects. Searching for files in a */proc* file system may require that the full path name be specified.

No text (**txt**) file descriptors are displayed for Linux processes. All entries for files other than the current working directory, the root directory, and numerical file descriptors are labeled **mem** descriptors.

Lsof can't search for Tru64 UNIX named pipes by name, because their kernel implementation of lstat(2) returns an improper device number for a named pipe.

Lsof can't report fully or correctly on HP-UX 9.01, 10.20, and 11.00 locks because of insufficient access to kernel data or errors in the kernel data. See the *lsof* FAQ (The **FAQ** section gives its location.) for details.

The AIX SMT file type is a fabrication. It's made up for file structures whose type (15) isn't defined in the AIX */usr/include/sys/file.h* header file. One way to create such file structures is to run X clients with the DISPLAY variable set to ''':0.0'''.

The **+|-f***[cfn]* option is not supported under /proc-based Linux *lsof*, because it doesn't read kernel structures from kernel memory.

ENVIRONMENT

Lsof may access these environment variables.

LANG — defines a language locale. See *setlocale*(3) for the names of other variables that can be used in place of LANG - e.g., LC_ALL, LC_TYPE, etc.

LSOFDEVCACHE — defines the path to a device cache file. See the **DEVICE CACHE PATH FROM AN ENVIRONMENT VARIABLE** section for more information.

LSOFPERSDCPATH — defines the middle component of a modified personal device cache file path. See the **MODIFIED PERSONAL DEVICE CACHE PATH** section for more information.

FAQ

Frequently-asked questions and their answers (an FAQ) are available in the *00FAQ* file of the *lsof* distribution.

That latest version of the file is found at:

https://github.com/lsof-org/lsof/blob/master/00FAQ

FILES

/dev/kmem	kernel virtual memory device
/dev/mem	physical memory device
/dev/swap	system paging device
.lsof_hostname	*lsof's* device cache file (The suffix, *hostname*, is the first component of the host's name returned by *gethostname*(2).)

AUTHORS

Lsof was written by Victor A.Abell <abe@purdue.edu> of Purdue University. Since version 4.93.0, the lsof-org team at GitHub maintains lsof. Many others have contributed to *lsof*. They're listed in the *00CREDITS* file of the *lsof* distribution.

DISTRIBUTION

The latest distribution of *lsof* is available at

https://github.com/lsof-org/lsof/releases

SEE ALSO

Not all the following manual pages may exist in every UNIX dialect to which *lsof* has been ported.

access(2), awk(1), crash(1), fattach(3C), ff(1), fstat(8), fuser(1), gethostname(2), isprint(3), kill(1), localtime(3), lstat(2), modload(8), mount(8), netstat(1), ofiles(8L), open(2), perl(1), ps(1), readlink(2), setlocale(3), stat(2), strftime(3), time(2), uname(1).