使用 lsof 查找打开的文件

通过查看打开的文件，了解更多关于系统的信息。了解应用程序打开了哪些文件或者哪个应用程序打开了特定的文件，作为系统管理员，这将使得您能够作出更好的决策。例如，您不应该卸载具有打开文件的文件系统。使用 lsof，您可以检查打开的文件，并根据需要在卸载之前中止相应的进程。同样地，如果您发现了一个未知的文件，那么可以找出到底是哪个应用程序打开了这个文件。

Sean A. Walberg (sean@ertw.com), 高级网络工程师

2006 年 8 月 21 日

内容

1 口评论

在 UNIX® 环境中，文件无处不在，这便产生了一句格言："任何事物都是文件"。通过文件不仅仅可以访问常规数据，通常还可以访问网络连接和硬件。在有些情况下，当您使用 ls 请求目录清单时，将出现相应的条目。在其他情况下，如传输控制协议 (TCP) 和用户数据报协议 (UDP) 套接字，不存在相应的目录清单。但是在后台为该应用程序分配了一个*文件描述符*，无论这个文件的本质如何，该文件描述符为应用程序与基础操作系统之间的交互提供了通用接口。

因为应用程序打开文件的描述符列表提供了大量关于这个应用程序本身的信息，所以能够查看这个列表将是很有帮助的。完成这项任务的实用程序称为 lsof，它对应于"list open files"（列出打开的文件）。几乎在每个 UNIX 版本中都有这个实用程序，但奇怪的是，大多数供应商并没有将其包含在操作系统的初始安装中。要获取更多关于 lsof 的信息，请参见参考资料部分。

lsof 简介

只需输入 lsof 就可以生成大量的信息，如清单 1 所示。因为 lsof 需要访问核心内存和各种文件，所以必须以 root 用户的身份运行它才能够充分地发挥其功能。

清单 1. lsof 的示例输出

```
bash-3.00# lsof
COMMAND    PID   USER   FD   TYPE        DEVICE SIZE/OFF      NODE NAME
sched        0   root  cwd   VDIR         136,8     1024         2 /
init         1   root  cwd   VDIR         136,8     1024         2 /
init         1   root  txt   VREG         136,8    49016      1655 /sbin/init
init         1   root  txt   VREG         136,8    51084      3185 /lib/libuutil.so.1
vi        2013   root    3u  VREG         136,8        0      8501 /var/tmp/ExXDaO7d
...
```

每行显示一个打开的文件，除非另外指定，否则将显示所有进程打开的所有文件。Command、PID 和 User 列分别表示进程的名称、进程标识符 (PID) 和所有者名称。Device、SIZE/OFF、Node 和 Name 列涉及到文件本身的信息，分别表示指定磁盘的名称、文件的大小、索引节点（文件在磁盘上的标识）和该文件的确切名称。根据 UNIX 版本的不同，可能将文件的大小报告为应用程序在文件中进行读取的当前位置（偏移量）。清单 1 来自一台可以报告该信息的 Sun Solaris 10 计算机，而 Linux® 没有这个功能。

FD 和 Type 列的含义最为模糊，它们提供了关于文件如何使用的更多信息。FD 列表示文件描述符，应用程序通过文件描述符识别该文件。Type 列提供了关于文件格式的更多描述。我们来具体研究一下文件描述符列，清单 1 中出现了三种不同的值。cwd 值表示应用程序的当前工作目录，这是该应用程序启动的目录，除非它本身对这个目录进行更改。txt 类型的文件是程序代码，如应用程序二进制文件本身或共享库，再比如本示例的列表中显示的 init 程序。最后，数值表示应用程序的文件描述符，这是打开该文件时返回的一个整数。在清单 1 输出的最后一行中，您可以看到用户正在使用 vi 编辑 /var/tmp/ExXDaO7d，其文件描述符为 3。u 表示该文件被打开并处于读取/写入模式，而不是只读 (r) 或只写 (w) 模式。有一点不是很重要但却很有帮助，初始打开每个应用程序时，都具有三个文件描述符，从 0 到 2，分别表示标准输入、输出和错误流。正因为如此，大多数应用程序所打开的文件的 FD 都是从 3 开始。

与 FD 列相比，Type 列则比较直观。根据具体操作系统的不同，您会发现将文件和目录称为 REG 和 DIR（在 Solaris 中，称为 VREG 和 VDIR）。其他可能的取值为 CHR 和 BLK，分别表示字符和块设备；或者 UNIX、FIFO 和 IPv4，分别表示 UNIX 域套接字、先进先出 (FIFO) 队列和网际协议 (IP) 套接字。

转到 /proc 目录

尽管与使用 lsof 没有什么直接的关系，但对 /proc 目录进行简要的介绍是有必要的。/proc 是一个目录，其中包含了反映内核和进程树的各种文件。这些文件和目录并不存在于磁盘中，因此当您对这些文件进行读取和写入时，实际上是在从操作系统本身获取相关信息。大多数与 lsof 相关的信息都存储于以进程的 PID 命名的目录中，所以 /proc/1234 中包含的是 PID 为 1234 的进程的信息。

在 /proc 目录的每个进程目录中存在着各种文件，它们可以使得应用程序简单地了解进程的内存空间、文件描述符列表、指向磁盘上的文件的符号链接和其他系统信息。lsof 实用程序使用该信息和其他关于内核内部状态的信息来产生其输出。稍后我将把 lsof 的输出与 /proc 目录中的信息联系起来。

常见用法

前面，我向您介绍了如何简单地运行不带任何参数的 lsof，以便显示关于每个进程所打开的文件的信息。本文余下的部分将重点关注如何使用 lsof 来显示所需的信息以及如何正确地对其进行解释。

查找应用程序打开的文件

lsof 常见的用法是查找应用程序打开的文件的名称和数目。您可能想尝试找出某个特定应用程序将日志数据记录到何处，或者正在跟踪某个问题。例如，UNIX 限制了进程能够打开文件的数目。通常这个数值很大，所以不会产生问题，并且在需要时，应用程序可以请求更大的值（直到某个上限）。如果您怀疑应用程序耗尽了文件描述符，那么可以使用 lsof 统计打开的文件数目，以进行验证。

要指定单个进程，可以使用 -p 参数，后面加上该进程的 PID。因为这样做不仅会返回该应用程序所打开的文件，还会返回共享库和代码，所以通常需要对输出进行筛选。要完成此任务，可以使用 -d 标志根据 FD 列进行筛选，使用 -a 标志表示两个参数都必须满足 (AND)。如果没有 -a 标志，缺省的情况是显示匹配任何一个参数 (OR) 的文件。清单 2 显示了 sendmail 进程打开的文件，并使用 txt 对这些文件进行筛选。

清单 2. 带有 PID 筛选器并进行 txt 文件描述符筛选的 lsof 输出

```
sh-3.00# lsof -a -p 605 -d ^txt
COMMAND  PID USER   FD   TYPE  DEVICE SIZE/OFF     NODE NAME
sendmail 605 root  cwd   VDIR  136,8     1024    23554 /var/spool/mqueue
sendmail 605 root    0r  VCHR  13,2           6815752 /devices/pseudo/mm@0:null
sendmail 605 root    1w  VCHR  13,2           6815752 /devices/pseudo/mm@0:null
sendmail 605 root    2w  VCHR  13,2           6815752 /devices/pseudo/mm@0:null
sendmail 605 root    3r  DOOR           0t0       58
  /var/run/name_service_door(door to nscd[81]) (FA:->0x30002b156c0)
sendmail 605 root    4w  VCHR  21,0          11010052
                 /devices/pseudo/log@0:conslog->LOG
sendmail 605 root    5u  IPv4 0x300010ea640      0t0      TCP *:smtp (LISTEN)
sendmail 605 root    6u  IPv6 0x3000431c180      0t0      TCP *:smtp (LISTEN)
sendmail 605 root    7u  IPv4 0x300046d39c0      0t0      TCP *:submission (LISTEN)
sendmail 605 root    8ww VREG         281,3       32  8778600 /var/run/sendmail.pid
```

清单 2 为 lsof 指定了三个参数。第一个是 -a，它表示当所有的参数都为真时，才显示这个文件。第二个参数是 -p 605，它限制仅输出 PID 为 605 的进程，可以通过 ps 命令获取这个信息。最后一个参数 -d ^txt，它表示筛选出其中 txt 类型的记录（脱字符号 [^] 表示排除）。

清单 2 的输出提供了关于进程行为的信息。如 cwd 行所示，该应用程序的工作目录为 /var/spool/mqueue。文件描述符 0、1 和 2 分配给了 /dev/null（Solaris 大量使用符号链接，所以这里显示了相应的伪设备）。FD 3 是一个 Solaris 门（高速远程过程调用 (RPC) 接口），以只读模式打开。FD 4 中的内容比较有趣，因为它是一个字符设备的只读句柄，实质上是 /dev/log。从这个文件中，您可以收集该应用程序向 UNIX syslog 守护进程进行的记录，所以 /etc/syslog.conf 规定了日志文件的位置。

作为一个网络应用程序，sendmail 对网络端口进行监听。文件描述符 5、6 和 7 可以告诉您，该应用程序正以 IPv4 和 IPv6 模式监听简单邮件传输协议 (SMTP) 端口，并以 IPv4 模式监听提交端口。最后一个文件描述符是只写的，并且指向 /var/run/sendmail.pid。FD 列中的大写 W 表示该应用程序具有对整个文件的写锁。该文件用于确保每次只能打开一个应用程序实例。

查找打开某个文件的应用程序

在其他情况下，您有一个文件或目录，并且需要知道哪个应用程序控制了该文件（打开了该文件）。清单 2 显示了由 sendmail 进程打开了 /var/run/sendmail.pid。如果您不知道这个信息，那么在给定文件名的情况下，lsof 可以提供该信息。清单 3 显示了相应的输出。

清单 3. 要求 lsof 显示关于某个文件的信息

```
bash-3.00# lsof /var/run/sendmail.pid
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
sendmail 605 root    8ww VREG  281,3       32 8778600 /var/run/sendmail.pid
```

正如输出所示，进程 sendmail（PID 为 605）控制了文件 /var/run/sendmail.pid，并且通过排它锁打开该文件以便进行写入。如果出于某种原因，您需要删除这个文件，那么正确的做法是中止该进程，而不是直接删除这个文件。否则，这个守护进程下次可能无法正常启动，或者可能稍后会启动另一个实例，从而导致争用。

有时您只知道在文件系统的某处打开了文件。在卸载文件系统时，如果该文件系统中有任何打开的文件，那么操作将会失败。通过指定装入点的名称，您可以使用 lsof 显示一个文件系统中所有打开的文件。清单 4 显示了如何尝试卸载 /export/home，然后使用 lsof 找出谁在使用该文件系统。

清单 4. 使用 lsof 找出谁在使用文件系统

```
bash-3.00# umount /export/home
umount: /export/home busy
bash-3.00# lsof /export/home
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
bash    1943 root  cwd   VDIR  136,7     1024    4 /export/home/sean
bash    2970 sean  cwd   VDIR  136,7     1024    4 /export/home/sean
ct      3030 sean  cwd   VDIR  136,7     1024    4 /export/home/sean
ct      3030 sean    1w  VREG  136,7        0   25 /export/home/sean/output
```

在这个示例中，用户 sean 正在其 home 目录中进行一些操作。有两个 bash（一种 Shell）实例正在运行，并且当前目录设置为 sean 的 home 目录。还有一个名为 ct 的应用程序正运行于相同的目录，并且其标准输出（文件描述符 1）重定向到一个名为 output 的文件。要成功地卸载 /export/home，应该在通知用户以确保情况正常之后，中止这些进程。

这个示例说明了应用程序的当前工作目录非常重要，因为它仍保持着文件资源，并且可以防止文件系统被卸载。这就是为什么大部分守护进程（后台进程）将它们的目录更改为根目录、或服务特定的目录（如 sendmail 示例中的 /var/spool/mqueue）的原因，以避免该守护进程阻止卸载不相关的文件系统。如果 sendmail 从 /export/home/sean 目录启动，并且没有将其目录更改为 /var/spool/mqueue，那么在卸载 /export/home 前必须中止它。

如果您对非装入点目录中打开的文件感兴趣，那么必须通过 +d 或 +D 指定该目录的名称，具体使用其中的哪一个标志取决于您需要递归到子目录（+D）或者不需要递归到子目录（+d）。例如，要查看 /export/home/sean 中所有打开的文件，可以使用 lsof +D /export/home/sean。在前面的示例中，相关的目录是一个装入点，而这里与前面的示例存在细微的差别，并且限制了 lsof 和内核之间的交互。这还会引起潜在的问题，即 lsof /export/home 与 lsof /export/home/（请注意尾部的斜杠）有所区别。第一种方式可以正常工作，因为它指向了装入点。第二种方式不会生成任何输出，因为它指向了目录。如果您在 Shell 中使用 Tab 键自动完成命令，那么可能碰到这个问题，其中会帮助您添加结尾的斜杠。在这种情况下，您可以删除这个斜杠或者使用 +D 指定目录。前者是首选的方法，因为与指定任意的目录相比，其执行速度更快。

不常见的用法

在前面的部分中，我们研究了 lsof 的基本用法，即显示打开的文件和控制它们的进程之间的关系。当您想对系统进行一些烦琐的操作，而又不希望破坏别人重要的文档时，这种方法很有帮助。您还可以使用相同的方法执行一些高难度的 UNIX 操作。

恢复删除的文件

当 UNIX 计算机受到入侵时，常见的情况是日志文件被删除，以掩盖攻击者的踪迹。管理错误也可能导致意外删除重要的文件，比如在清理旧日志时，意外地删除了数据库的活动事务日志。有时可以恢复这些文件，并且 lsof 可以为您提供帮助。

当进程打开了某个文件时，只要该进程保持打开该文件，即使将其删除，它依然存在于磁盘中。这意味着，进程并不知道文件已经被删除，它仍然可以向打开该文件时提供给它的文件描述符进行读取和写入。除了该进程之外，这个文件是不可见的，因为已经删除了其相应的目录条目。

前面曾在转到 /proc 目录部分中说过，通过在适当的目录中进行查找，您可以访问进程的文件描述符。在随后的内容中，您看到了 lsof 可以显示进程的文件描述符和相关的文件名。您能明白我的意思吗？

但愿它真的这么简单！当您向 lsof 传递文件名时，比如在 lsof /file/I/deleted 中，它首先使用 stat() 系统调用获得有关该文件的信息，不幸的是，这个文件已经被删除。在不同的操作系统中，lsof 可能可以从核心内存中捕获该文件的名称。清单 5 显示了一个 Linux 系统，其中意外地删除了 Apache 日志，我正使用 grep 工具查找是否有人打开了该文件。

清单 5. 在 Linux 中使用 lsof 查找删除的文件

```
# lsof | grep error_log
httpd      2452     root    2w      REG       33,2      499    3090660
          /v a r / l o g / h t t p d / e r r o r _ l o g  ( d e l e t e d )
httpd      2452     root    7w      REG       33,2      499    3090660
          /v a r / l o g / h t t p d / e r r o r _ l o g  ( d e l e t e d )
... more httpd processes ...
```

在这个示例中，您可以看到 PID 2452 打开文件的文件描述符为 2（标准错误）和 7。因此，可以在 /proc/2452/fd/7 中查看相应的信息，如清单 6 所示。

清单 6. 通过 /proc 查找删除的文件

```
# cat /proc/2452/fd/7
[Sun Apr 30 04:02:48 2006] [notice] Digest: generating secret for digest authentication
[Sun Apr 30 04:02:48 2006] [notice] Digest: done
[Sun Apr 30 04:02:48 2006] [notice] LDAP: Built with OpenLDAP LDAP SDK
```

Linux 的优点在于，它保存了文件的名称，甚至可以告诉我们它已经被删除。在遭到破坏的系统中查找相关内容时，这是非常有用的内容，因为攻击者通常会删除日志以隐藏他们的踪迹。Solaris 并不提供这些信息。然而，我们知道 httpd 守护进程使用了 error_log 文件，所以可以使用 ps 命令找到这个 PID，然后可以查看这个守护进程打开的所有文件。

清单 7. 在 Solaris 中查找删除的文件

```
# lsof -a -p 8663 -d ^txt
COMMAND  PID   USER   FD   TYPE        DEVICE SIZE/OFF    NODE NAME
httpd   8663 nobody  cwd   VDIR         136,8     1024       2 /
httpd   8663 nobody   0r  VCHR          13,2          6815752 /devices/pseudo/mm@0:null
httpd   8663 nobody   1w  VCHR          13,2          6815752 /devices/pseudo/mm@0:null
httpd   8663 nobody   2w  VREG         136,8      185  145465 / (/dev/dsk/c0t0d0s0)
httpd   8663 nobody   4r  DOOR                0t0       58 /var/run/name_service_door
          ( d o o r   t o   n s c d [ 8 1 ] )   ( F A : - > 0 x 3 0 0 0 2 b 1 5 6 c 0 )
httpd   8663 nobody  15w  VREG         136,8      185  145465 / (/dev/dsk/c0t0d0s0)
httpd   8663 nobody  16u  IPv4 0x300046d27c0      0t0      TCP *:80 (LISTEN)
httpd   8663 nobody  17w  VREG         136,8        0  145466
                                                      /var/apache/logs/access_log
httpd   8663 nobody  18w  VREG         281,3        0 9518013 /var/run (swap)
```

我使用 -a 和 -d 参数对输出进行筛选，以排除代码程序段，因为我知道需要查找的是哪些文件。NAME 列显示出，其中的两个文件（FD 2 和 15）使用磁盘名代替了文件名，并且它们的类型为 VREG（常规文件）。在 Solaris 中，删除的文件将显示文件所在的磁盘的名称。通过这个线索，就可以知道该 FD 指向一个删除的文件。实际上，查看 /proc/8663/fd/15 就可以得到所要查找的数据。

如果可以通过文件描述符查看相应的数据，那么您就可以使用 I/O 重定向将其复制到文件中，如 cat /proc/8663/fd/15 > /tmp/error_log 。此时，您可以中止该守护进程（这将删除 FD，从而删除相应的文件），将这个临时文件复制到所需的位置，然后重新启动该守护进程。

对于许多应用程序，尤其是日志文件和数据库，这种恢复删除文件的方法非常有用。正如您所看到的，有些操作系统（以及不同版本的 lsof）比其他的系统更容易查找相应的数据。

查找网络连接

网络连接也是文件，这意味着可以使用 lsof 获得关于它们的信息。您曾在清单 2 中看到过这样的示例。该示例假设您已经知道 PID，但是有时候并非如此。如果您只知道相应的端口，那么可以使用 -i 参数利用套接字信息进行搜索。清单 8 显示了对 TCP 端口 25 的搜索。

清单 8. 查找监听端口 25 的进程

```
# lsof -i :25
COMMAND  PID USER   FD   TYPE        DEVICE SIZE/OFF NODE NAME
sendmail 605 root    5u  IPv4 0x300010ea640      0t0  TCP *:smtp (LISTEN)
sendmail 605 root    6u  IPv6 0x3000431c180      0t0  TCP *:smtp (LISTEN)
```

需要以 protocol:@ip:port 的形式向 lsof 实用程序传递相关信息，其中的 protocol 为 TCP 或 UDP（可以使用 4 或 6 作为前缀，表示 IP 的版本），IP 为可解析的名称或 IP 地址，而 port 为数字或表示该服务的名称（来自 /etc/services）。需要一个或多个元素（端口、IP、协议）。在清单 8 中，:25 表示端口 25。输出显示，进程 605 正在使用 IPv6 和 IPv4 监听端口 25。如果您对 IPv4 不感兴趣，那么可以将筛选器改为 6:25，以表示监听端口 25 的 IPv6 套接字，或者直接使用 6 表示所有的 IPv6 连接。

除了显示出这些守护进程正在监听的对象，lsof 还可以发现发生的连接，同样是使用 -i 参数。清单 9 显示了搜索与 192.168.1.10 之间的所有连接。

清单 9. 搜索活动的连接

```
# lsof -i @192.168.1.10
COMMAND  PID USER   FD   TYPE        DEVICE  SIZE/OFF NODE NAME
sshd    1934 root    6u  IPv6 0x300046d21c0 0t13036O8  TCP sun:ssh->linux:40379
                              (   E   S   T   A   B   L   I   S   H   E   D   )
sshd    1937 root    4u  IPv6 0x300046d21c0 0t13036O8  TCP sun:ssh->linux:40379
                              (   E   S   T   A   B   L   I   S   H   E   D   )
```

在这个示例中，sun 和 linux 之间有两个 IPv6 连接。对其进行更仔细的研究可以看出，这些连接来自于两个不同的进程，但它们却是相同的，这是因为两台主机是相同的，并且端口也是相同的（ssh 和 40379）。这是由于进入主进程的连接分叉出一个处理程序，并将该套接字传递给它。您还可以看到，名为 sun 的计算机正在使用端口 22 (ssh)，而 linux 具有端口 40379。这表示，sun 是该连接的接收者，因为它关联于该服务的已知端口。40379 是源或临时端口，并且仅对这个连接有意义。

因为，至少在 UNIX 中，套接字是另一类文件，所以 lsof 可以获得关于这些连接的详细信息，并找出谁对它们负责。

结束语

UNIX 大量使用了文件。作为系统管理员，lsof 允许您对核心内存进行查看，以找出系统当前如何使用这些文件。lsof 最简单的用法可以告诉您哪些进程打开了哪些文件，以及哪些文件由哪些进程打开。在收集关于应用程序工作情况的信息时，或在进行某些可能损坏数据的操作前确保文件未被使用时，这一点特别重要 lsof 更高级的用法可以帮助您查找删除的文件，并获得关于网络连接的信息。这是一个功能强大的工具，它几乎可以用于任何地方。

参考资料

学习

- 您可以参阅本文在 developerWorks 全球站点上的 英文原文 。
- Introduction to reverse engineering software：这个站点使用 lsof 和其他的 UNIX 工具，以找出关于给定二进制文件的所有信息。
- lsof 的 man 页面：如果您发现控制台版本的内容不大容易读懂，那么这个页面将提供非常有用的内容。
- AIX and UNIX：想了解更多内容吗？developerWorks 的 AIX and UNIX 专区提供数百篇关于 AIX 和 UNIX 的文章以及入门级、中级和高级教程，将让您打开眼界。
- developerWorks 技术事件与网络广播：跟踪最新的 developerWorks 技术事件与网络广播。
- 技术讲座：收听技术讲座并保持与 IBM 技术专家同步。

获得产品和技术

- IBM 试用软件：使用 IBM 软件开发您的下一个项目，可直接从 developerWorks 下载这些试用软件。

讨论

- 参与 "AIX and UNIX"论坛、developerWorks 博客，并加入 developerWorks 社区。

条评论

请 登录 或 注册 后发表评论。

添加评论:

注意: 评论中不支持 HTML 语法

有新评论时提醒我

剩余 1000 字符

发布

共有评论 (1)

Thanks a lot !

由 reetsee 于 2015年10月30日发布

报告滥用

回页首