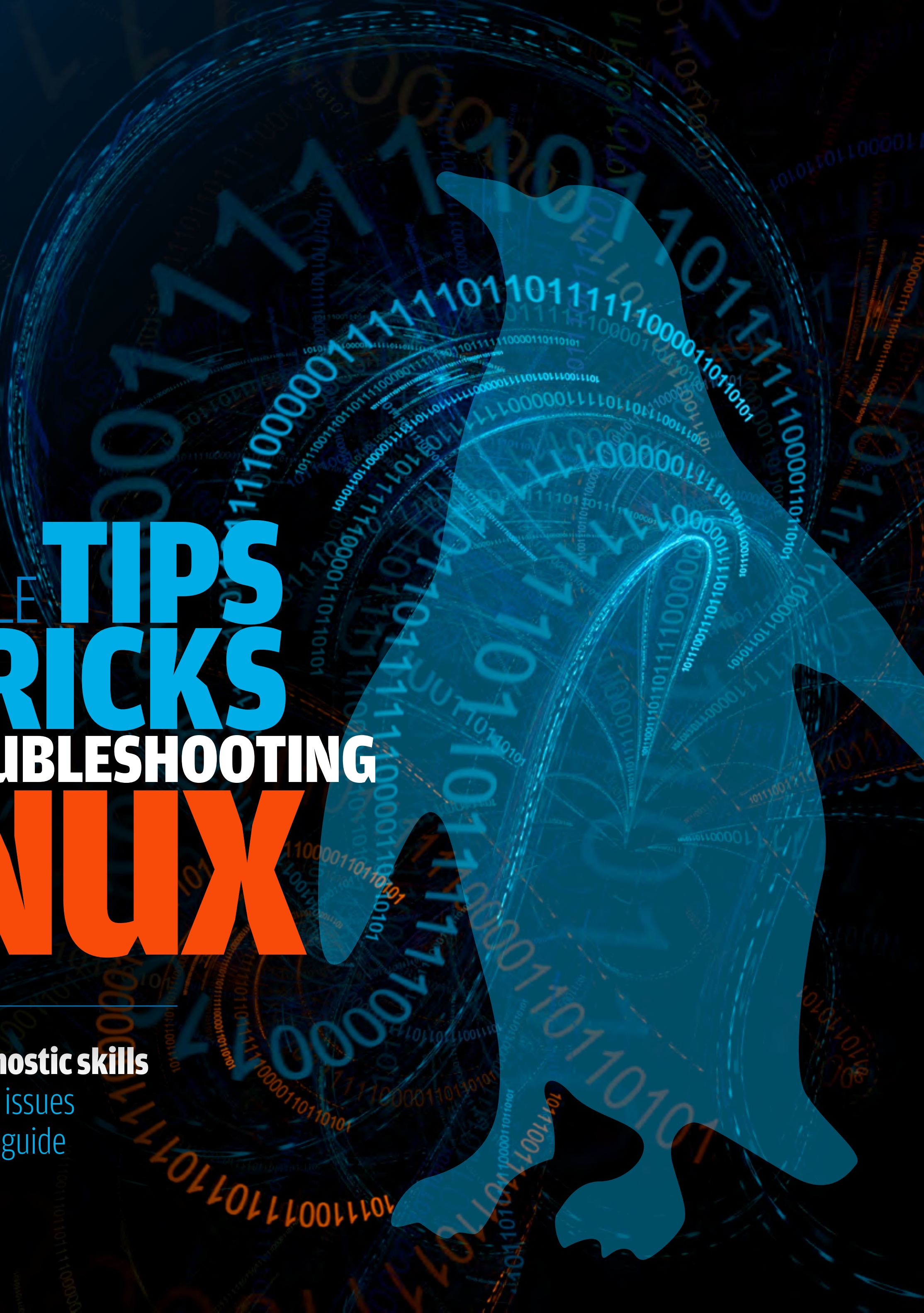
NETWORK WORLD
FROM IDG

THE CONNECTED ENTERPRISE

INSIDER EXCLUSIVE



INVALUABLE TIPS + TRICKS FOR TROUBLESHOOTING LINUX

Hone your diagnostic skills for handling Linux issues with this essential guide

NetworkWorld.com

Editor's NOTE

TROUBLESHOOTING ANYTHING comes with its own set of challenges, and in the case of Linux and Unix, it's helpful to have a practical guide to the tools available to make the job easier. This is such a guide to introduce Linux/Unix admins to some essential commands that can make their lives easier when problems crop up.

The "list open files" or lsof command sounds straightforward enough, but its use as a troubleshooting tool may not be as apparent. For example, if an unknown process has a number of files open, knowing which ones they are can help determine whether the process is legitimate.

The first article in this guide explains many options for using lsof, so many, in fact, that we've include a second article with recommendations on how to keep them all straight. This cheat sheet recommends some aliases for the commands and a framework that you might use to create more.

A less technical checklist of questions to ask yourself when diagnosing problems is the subject of our third article. These will help impose some order on the chaos that some problems create.

Dealing with Oracle databases is a common admin task, and knowing a bit about them and how to test connections to them is valuable tool described in the fourth article. If the connection is the problem, this can rescue you from calling in the database admin.

We round out our guide with instruction on the commands needed to manage and partition disks that first offers a primer on how disks work and the fundamentals of disk management.

Enjoy! ◆

TIM GREENE, *executive editor, Network World*

INSIDE

SANDRA HENRY-STOCKER *has been administering Unix systems for more than 30 years. You can contact her at bugfarm@gmail.com.*

UNIX COMMANDS: TROUBLESHOOTING WITH LSOF

There's more to the lsof command than you might imagine. Check out all the ways that it can be used to help you with your troubleshooting. **BY SANDRA HENRY-STOCKER**

To comment on this story, visit Network World's Facebook page.

IF YOU'VE NEVER USED the lsof command or used it only for one specific purpose, you might be delighted to learn how many ways the lsof (list open files) command can help you manage your servers. lsof is the Unix/Linux command that allows you to list open files or identify the processes that particular files have open. Handy for evaluating system security as well as troubleshooting, lsof features a large range of options that allow it to be used in numerous ways — sometimes even surpassing the ps command for looking at processes and the netstat command for examining network interfaces.

What are open files?

For starters, let's consider what open files are and why you might be curious about them. Open files are files that some process is using. That process might be a command that you are running or an application that is running on a server you manage. The open files might include data files and libraries that supply shared routines. Many files are opened every time you log in. You might be surprised at how many. If you're curious about how many files you have open right now, try this command:

```
$ lsof -u `whoami` | wc -1
55
```

And, if you've ever heard anyone say that for Unix, everything is a file, you might not be too surprised to learn that lsof works with things — like network interfaces — that most of us don't generally think of as files.

Why do you care?

Sometimes you might want to know about open files because you try to remove a file and discover that it's in use. Maybe it's filling up your disk space like a desperate shopper at a dollar store. You want to know what process has the file open so that you can stop it and empty the file. At other times, you might want to know what some suspicious process is doing, and examining the files that it has open can provide valuable information.

How does lsof work?

When used without options, lsof lists all files that are open (in use) on your system. If you run the lsof as yourself, you will get a long listing, but the output will include a lot of permission denied messages – many representing open files in the /proc file system that you're not allowed to see. Run the command as root and you'll see more output and all the data.

NOTE: *The command output below (as well as the output for most of the commands shown in this post) has been truncated to keep this post from turning into a virtual ebook. The ellipsis (...) at the end of each patch of output is meant to indicate where output was omitted.*

```
$ lsof
COMMAND   PID  USER   FD    TYPE     DEVICE       SIZE/OFF    NODE NAME
init        1  root   cwd   unknown  /proc/1/cwd  (readlink:  Permission denied)
init        1  root   rtd   unknown  /proc/1/root (readlink: Permission denied)
init        1  root   txt   unknown  /proc/1/exe  (readlink:  Permission denied)
...
```

```
$ sudo lsof
COMMAND   PID  USER   FD    TYPE     DEVICE       SIZE/OFF    NODE NAME
init        1  root   cwd   DIR      202,1        4096        2 /
init        1  root   rtd   DIR      202,1        4096        2 /
init        1  root   txt   REG      202,1        150360      397362 /sbin/init
init        1  root   DEL   REG      202,1                    396485 /lib64/libnss_files-2.17.so
...
```

What else is there?

A quick glimpse at the rather length lsof man page will show you that we've only skimmed the surface. The lsof command has an extensive list of options.

```
lsof [ -?abChlnNOPRtUvVX ] [ -A A ] [ -c c ] [ +c c ] [ +|-d d ] [ +|-D
D ] [ +|-e s ] [ +|-f [cfgGn] ] [ -F [f] ] [ -g [s] ] [ -i [i] ] [ -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]
```

We're only going to look at some of the more useful ones in this post.

To get started with all of these options, you should know that, if you use more than one, they get ORed together. In other words, you get a listing that combines the results of the options you specified. You can choose instead to use an option which ANDs your specifications together. In other words, you will see only the files or processes, etc. that match all of the options you specify. To use the AND option, add the -a option (there's an example command below) to your command.

For a quick review of the implications of AND and OR, if you ask a programmer for a peanut butter AND jelly sandwich, you'll probably end up with just two slices of bread — because the toppings can't be both peanut butter and jelly at the same time. The logic goes something like this: if INGREDIENT = "peanut butter" AND INGREDIENT = "jelly", SANDWICH-TOPPINGS = NULL.

Helpful lsof commands

The sample lsof commands below help illustrate some of the more useful things that you can do with the command. Again, most of the output is truncated to keep the length of this post reasonable.

- **The sample lsof command below will list all processes that have a particular file open:**

```
$ sudo lsof /lib64/libcrypto.so.1.0.1k
COMMAND    PID   USER    FD    TYPE    DEVICE     SIZE/OFF    NODE NAME
ntpd       2148  ntp     mem   REG     202,1      1971624     399258 /lib64/libcrypto.so.1.0.1k
sendmail   2163  root    mem   REG     202,1      1971624     399258 /lib64/libcrypto.so.1.0.1k
sendmail   2170  smmsp   mem   REG     202,1      1971624     399258 /lib64/libcrypto.so.1.0.1k
sshd      13774  root    mem   REG     202,1      1971624     399258 /lib64/libcrypto.so.1.0.1k
sshd      13776  froggy  mem   REG     202,1      1971624     399258 /lib64/libcrypto.so.1.0.1k
sshd      22551  root    mem   REG     202,1      1971624     399258 /lib64/libcrypto.so.1.0.1k
...
```

This command will list all processes that have files within a particular directory open:

```
$ sudo lsof +D /lib64 | more
COMMAND    PID   USER    FD    TYPE    DEVICE    SIZE/OFF    NODE NAME
init         1   root    mem   REG     202,1     89312       396417 /lib64/libgcc_s-4.8.2-20140120.so.1
init         1   root    mem   REG     202,1     283184      397017 /lib64/libdbus-1.so.3.7.4
init         1   root    mem   REG     202,1     39936       397358 /lib64/libnih-dbus.so.1.0.0
init         1   root    mem   REG     202,1     101992      397360 /lib64/libnih.so.1.0.0
dhclient  1848   root    mem   REG     202,1     18712       396864 /lib64/libcap-ng.so.0.0.0
auditd    1889   root    mem   REG     202,1     89312       396417 /lib64/libgcc_s-4.8.2-20140120.so.1
auditd    1889   root    mem   REG     202,1     13224       396979 /lib64/libkeyutils.so.1.5
...
```

In this command, we look at files opened by bash:

```
$ sudo lsof -c bash | more
COMMAND    PID   USER    FD    TYPE    DEVICE    SIZE/OFF    NODE NAME
bash     17811   froggy  cwd   DIR     202,1     4096        410779 /home/froggy
bash     17811   froggy  rtd   DIR     202,1     4096        2 /
bash     17811   froggy  txt   REG     202,1     898032      396531 /bin/bash
bash     17811   froggy  mem   REG     202,1     106065056   410803 /usr/lib/locale/locale-archive
bash     17811   froggy  mem   REG     202,1     58288       396484 /lib64/libnss_files-2.17.so
bash     17811   froggy  mem   REG     202,1     2107600     396466 /lib64/libc-2.17.so
bash     17811   froggy  mem   REG     202,1     19512       396472 /lib64/libdl-2.17.so
bash     17811   froggy  mem   REG     202,1     135616      396516 /lib64/libtinfo.so.5.7
bash     17811   froggy  mem   REG     202,1     160240      396459 /lib64/ld-2.17.so
bash     17811   froggy  mem   REG     202,1     26254       1596 /usr/lib64/gconv/gconv-modules.cache
bash     17811   froggy  0u    CHR     136,0     0t0         3 /dev/pts/0
bash     17811   froggy  1u    CHR     136,0     0t0         3 /dev/pts/0
bash     17811   froggy  2u    CHR     136,0     0t0         3 /dev/pts/0
bash     17811   froggy  255u  CHR     136,0     0t0         3 /dev/pts/0
```

Below, we do the same thing but use a substring instead of the full process name:

```
$ sudo lsof -c bas
$ sudo lsof -c bas
COMMAND    PID   USER      FD    TYPE    DEVICE    SIZE/OFF    NODE NAME
bash     17811   ec2-user  cwd   DIR     202,1     4096        410779 /home/froggy
bash     17811   ec2-user  rtd   DIR     202,1     4096        2 /
bash     17811   ec2-user  txt   REG     202,1     898032      396531 /bin/bash
bash     17811   ec2-user  mem   REG     202,1     106065056   410803 /usr/lib/locale
...                                                            /locale-archive
```

- **Next, we list open files for a particular process ID:**

```
$ sudo lsof -p 2178
COMMAND   PID   USER   FD    TYPE   DEVICE               SIZE/OFF   NODE NAME
crond     2178  root   cwd   DIR    202,1                4096       2 /
crond     2178  root   rtd   DIR    202,1                4096       2 /
crond     2178  root   txt   REG    202,1                64096      403907 /usr/sbin/crond
crond     2178  root   DEL   REG    202,1                           396485 /lib64/libnss_files-2.17.so
crond     2178  root   DEL   REG    202,1                           396511 /usr/lib/locale/locale-archive
crond     2178  root   DEL   REG    202,1                           396467 /lib64/libc-2.17.so
crond     2178  root   mem   REG    202,1                113112     396681 /lib64/libaudit.so.1.0.0
crond     2178  root   DEL   REG    202,1                           396473 /lib64/libdl-2.17.so
crond     2178  root   mem   REG    202,1                55928      398855 /lib64/libpam.so.0.83.1
crond     2178  root   mem   REG    202,1                126336     396609 /usr/lib64/libselinux.so.1
crond     2178  root   DEL   REG    202,1                           396460 /lib64/ld-2.17.so
crond     2178  root   0u    CHR    1,3                  0t0        5422 /dev/null
crond     2178  root   1u    CHR    1,3                  0t0        5422 /dev/null
crond     2178  root   2u    CHR    1,3                  0t0        5422 /dev/null
crond     2178  root   3u    REG    202,1                5          263404 /var/run/crond.pid
crond     2178  root   4u    unix   0xffff88003d754980   0t0        9655 socket
crond     2178  root   5r    0000   0,9                  0          5420 anon_inode
```

Using lsof to look at your network

- **We can also use lsof to look at network connections:**

```
$ sudo lsof -i
COMMAND   PID    USER      FD    TYPE   DEVICE     SIZE/OFF   NODE NAME
dhclient  1848   root      5u    IPv4   8573       0t0        UDP *:bootpc
ntpd      2148   ntp       16u   IPv4   9534       0t0        UDP *:ntp
ntpd      2148   ntp       17u   IPv4   9538       0t0        UDP localhost:ntp
ntpd      2148   ntp       18u   IPv4   9539       0t0        UDP 172.30.0.28:ntp
sendmail  2163   root      4u    IPv4   9613       0t0        TCP localhost:smtp
                                                              (LISTEN)
sshd      15350  root      3u    IPv4   17247327   0t0        TCP 172.30.0.28:ssh>
                                                              ip108.cable.shentel.net:37884
                                                              (ESTABLISHED)
sshd      15352  ec2-user  3u    IPv4   17247327   0t0        TCP 172.30.0.28:ssh>
                                                              ip108.cable.shentel.net:37884
                                                              (ESTABLISHED)
sshd      22551  root      3u    IPv4   32640      0t0        TCP *:ssh (LISTEN)
sshd      22551  root      4u    IPv6   32642      0t0        TCP *:ssh (LISTEN)
```

We can look for listening ports or established connections.

```
$ sudo lsof -i -sTCP:LISTEN
COMMAND    PID  USER   FD   TYPE  DEVICE    SIZE/OFF  NODE NAME
sendmail  2163  root   4u   IPv4  9613      0t0       TCP localhost:smtp (LISTEN)
sshd     22551  root   3u   IPv4  32640     0t0       TCP *:ssh (LISTEN)
sshd     22551  root   4u   IPv6  32642     0t0       TCP *:ssh (LISTEN)
```

```
$ sudo lsof -i -sTCP:ESTABLISHED
COMMAND    PID  USER    FD   TYPE  DEVICE    SIZE/OFF  NODE NAME
sshd     13060  root    3u   IPv4  17234339  0t0       TCP 172.30.0.28:ssh->
                                                       ip108.cable.shentel.net:37846 (ESTABLISHED)
sshd     13062  froggy  3u   IPv4  17234339  0t0       TCP 172.30.0.28:ssh->
                                                       ip108.cable.shentel.net:37846 (ESTABLISHED)
```

View IPv6 traffic only

```
$ sudo lsof -i 6
COMMAND    PID  USER   FD   TYPE  DEVICE    SIZE/OFF  NODE NAME
sshd     22551  root   4u   IPv6  32642     0t0       TCP *:ssh (LISTEN)
```

```
$ sudo lsof -iUDP
COMMAND    PID  USER   FD   TYPE  DEVICE    SIZE/OFF  NODE NAME
dhclient  1848  root   5u   IPv4  8573      0t0       UDP *:bootpc
ntpd      2148  ntp    16u  IPv4  9534      0t0       UDP *:ntp
ntpd      2148  ntp    17u  IPv4  9538      0t0       UDP localhost:ntp
ntpd      2148  ntp    18u  IPv4  9539      0t0       UDP 172.30.0.28:ntp
```

Or we can look at network connections for one particular source.

```
$ sudo lsof -i@172.30.0.28
COMMAND    PID  USER    FD   TYPE  DEVICE    SIZE/OFF  NODE NAME
ntpd      2148  ntp     18u  IPv4  9539      0t0       UDP 172.30.0.28:ntp
sshd     13060  root    3u   IPv4  17234339  0t0       TCP 172.30.0.28:ssh->
                                                       ip108.cable.shentel.net:37846 (ESTABLISHED)
sshd     13062  froggy  3u   IPv4  17234339  0t0       TCP 172.30.0.28:ssh->
                                                       ip108.cable.shentel.net:37846 (ESTABLISHED)
```

- **We can also do the same thing but only for one particular network connection.**

```
$ sudo lsof -u ec2-user -i @172.30.0.28
COMMAND   PID    USER      FD    TYPE  DEVICE    SIZE/OFF  NODE NAME
ntpd      2148   ntp       18u   IPv4  9539      0t0       UDP 172.30.0.28:ntp
sshd      15350  root      3u    IPv4  17247327  0t0       TCP 172.30.0.28:ssh->
                                                           ip108.cable.shentel.net:37884
                                                           (ESTABLISHED)
sshd      15352  ec2-user  cwd   DIR   202,1     4096      2 /
sshd      15352  ec2-user  rtd   DIR   202,1     4096      2 /
sshd      15352  ec2-user  txt   REG   202,1     617128    404007 /usr/sbin/sshd
...
```

Looking at files by user

- **This lsof command allows you to view open files for a particular user:**

```
$ sudo lsof -u froggy
COMMAND   PID    USER    FD    TYPE  DEVICE  SIZE/OFF  NODE NAME
sshd      13062  froggy  cwd   DIR   202,1   4096      2 /
sshd      13062  froggy  rtd   DIR   202,1   4096      2 /
sshd      13062  froggy  txt   REG   202,1   617128    404007 /usr/sbin/sshd
sshd      13062  froggy  DEL   REG   0,4     17234382  /dev/zero
sshd      13062  froggy  mem   REG   202,1   18640     398879 /lib64/security/pam_limits.so
sshd      13062  froggy  mem   REG   202,1   10264     398877 /lib64/security/pam_keyinit.so
sshd      13062  froggy  mem   REG   202,1   10288     398882 /lib64/security/pam_loginuid.so
sshd      13062  froggy  mem   REG   202,1   18736     398894 /lib64/security/pam_selinux.so
sshd      13062  froggy  mem   REG   202,1   38976     398532 /usr/lib64/libcrack.so.2.8.1
sshd      13062  froggy  mem   REG   202,1   14456     398863 /lib64/security/pam_cracklib.so
...
```

- **To view open files for all users except some particular user (in this case, root), use a ^ sign:**

```
$ sudo lsof -u ^root | more
COMMAND    PID   USER  FD    TYPE  DEVICE  SIZE/OFF  NODE NAME
dbus-daem  1935  dbus  cwd   DIR   202,1   4096      2 /
dbus-daem  1935  dbus  rtd   DIR   202,1   4096      2 /
dbus-daem  1935  dbus  txt   REG   202,1   403416    404014 /bin/dbus-daemon
```

```
COMMAND    PID   USER    FD    TYPE    DEVICE      SIZE/OFF    NODE NAME
dbus-daem  1935  dbus    DEL   REG     202,1                   396485 /lib64/libnss_files-2.17.so
dbus-daem  1935  dbus    DEL   REG     202,1                   396473 /lib64/libdl-2.17.so
dbus-daem  1935  dbus    DEL   REG     202,1                   396467 /lib64/libc-2.17.so
dbus-daem  1935  dbus    DEL   REG     202,1                   396497 /lib64/librt-2.17.so
dbus-daem  1935  dbus    DEL   REG     202,1                   396493 /lib64/libpthread-2.17.so
dbus-daem  1935  dbus    mem   REG     202,1       18712       396864 /lib64/libcap-ng.so.0.0.0
dbus-daem  1935  dbus    mem   REG     202,1       113112      396681 /lib64/libaudit.so.1.0.0
```

- **List process IDs for processes being run by a particular user:**

```
$ sudo lsof -t -u froggy
15352
15353
```

- **Kill all processes belonging to a particular user:**

```
$ sudo kill `lsof -t -u froggy`
```

Switching to AND

- **Use the -a option to AND your conditions together, keeping in mind that this limits the output to only that which matches all specified conditions:**

```
$ sudo lsof -i -a -p 2148

COMMAND    PID   USER    FD    TYPE    DEVICE      SIZE/OFF    NODE NAME
ntpd       2148  ntp     16u   IPv4    9534        0t0         UDP *:ntp
ntpd       2148  ntp     17u   IPv4    9538        0t0         UDP localhost:ntp
ntpd       2148  ntp     18u   IPv4    9539        0t0         UDP 172.30.0.28:ntp
...
```

Wrap up

The lsof command is a lot more useful and versatile than many Unix admins realize. It could come in handy for a lot of tasks that you've been addressing in other ways. Try out some of these commands and let me know which you like the most. And don't forget to make your own peanut butter and jelly sandwiches! ◆

Making Troubleshooting with Lsof Easier

The lsof command has so many options that you may not be making good use of it. **Let's look at ways that you can make it work better for you.** **BY SANDRA HENRY-STOCKER**

IN OUR LEAD STORY, we looked at a series of commands that used the lsof (list open files) command to provide information that can help with troubleshooting on the Unix systems you manage. Since lsof has such a huge collection of options, remembering which option to use for what sometimes makes the command hard to use as often or as effectively as you might like. So what we're doing today is looking at several ways to make the use of this very helpful tool a bit easier. We do that by creating useful aliases, by providing something of a "cheat sheet," and by deploying a number of lsof options in a script that makes educated guesses about what you're going after.

Using aliases

Both of the aliases below will list whatever files are open on your behalf when you are logged in. I suspect that few sysadmins will want to type "showmyopenfiles." It might be less of a problem to remember the lsof option or print out a cheat sheet. On the other hand, "showmine" would be somewhat ambiguous – my open files or my processes?

Note that most of these aliases require root privilege and assume that you have sudo privileges.

```
alias showmyopenfiles=’sudo lsof -u `whoami`’
alias showmine=’sudo lsof -u `whoami`’
```

Maybe “showfiles” ,“showmyfiles” or just “ofiles” would work better.

```
alias showfiles=’sudo lsof’
alias showmyfiles=’sudo lsof -u `whoami`’
alias ofiles=’sudo lsof -u `whoami`’
```

In the command shown below, we’re looking for the processes that have opened /usr/sbin/lsof — the lsof command itself.

```
$ showfiles /usr/sbin/lsof
COMMAND   PID   USER   FD    TYPE   DEVICE    SIZE/OFF   NODE NAME
lsof     27473  root   txt   REG    202,1     141048     407194 /usr/sbin/lsof
lsof     27474  root   txt   REG    202,1     141048     407194 /usr/sbin/lsof
```

Of course, getting overly clever with your aliases might make them harder to use than just going with basic lsof commands. Another option is to create aliases for the handful of options that you’re likely to use most of the time.

```
alias byuser=’sudo lsof -u’
alias bypid=’sudo lsof -p’
alias byfile=’sudo lsof’
alias byprogram=’sudo lsof -c’
```

Anyone using these aliases just has to remember to add the argument (username, PID, etc.).

In a similar way, you can set up aliases that show information for your network connections.

```
alias shownet=’sudo lsof -i’
alias showtcp=’sudo lsof -i tcp’
alias showudp=’sudo lsof -i udp’
```

Using a cheat sheet

Having a cheat sheet on hand with quick explanations of the lsof command’s options can also help you take advantage of its many features without having to memorize them. Simple explanations and sample commands seem to be the most helpful. Here’s an example:

What files are open?	lsof
What process has a particular file open?	lsof /path/to/the/file
What files in some directory are open?	lsof +D /path/to/the/dir
What files does some user have open?	lsof -u username
What files does a group of users have open?	lsof -u user1,user2
What files are open by process name?	lsof -c procname
What files are open by PID?	lsof -p 123
What files are open by other PIDs?	lsof -p ^123
Show network activity	lsof -i
What files are open by port?	lsof -i :25
	lsof -i :smtp
List PIDs	lsof -t
Show network activity for a user	lsof -a -u username -i
Show socket use	lsof -U
Show NFS activity	lsof -N

```
#!/bin/bash

if [ $# == 0 ]; then
  echo “USAGE: $0 <what>”
  echo “Example: $0 procid”
fi

case $1 in
  [0-9]*.[0-9]*.[0-9]*.[0-9]*) sudo lsof -i@$1;;
  [0-99999]*) lsof -p $1;;
  net)      sudo lsof -i;;
  [a-z]*)    who | grep $1;
        if [ $? == 0 ]; then
          sudo lsof -u $1
        else
          if [ -e $1 ]; then
            sudo lsof $1
          else
            sudo lsof -i :$1
          fi
        fi;;
  /*)      if [ -f $1 ]; then
          sudo lsof $1
        fi;;
  *)       echo “Sorry – target not recognized”;;
esac
```

Using a script

You can also simplify use of the lsof command by creating a script. The one to the left tries to determine what you’re looking for by evaluating the argument that you provide. For example, if you enter an IP address, it assumes that you want to see network activity for that particular IP. Feel free to modify it to better represent your own troubleshooting focus.

The items in the case statement need to be ordered in such a way that the more restrictive choices come first (e.g., IP addresses before numbers). This script looks for IP addresses, numbers (which it assumes are PIDs), strings (which it first tries to identify as usernames, then looks for a matching local file, and then tries it as a port. If the argument starts with a /, it assumes it’s a file.

Wrap Up

There are many ways to make routine use of the lsof command easier and more likely. I hope some of the options presented in this article will prove to be useful. ◆

When Your System Breaks: Unix Troubleshooting Basics

Questions sysadmins should ask and precautions they should take when troubleshooting. **BY SANDRA HENRY-STOCKER**

GENERALLY NOT TAUGHT in any formal classes, troubleshooting is one of the things that most of us end up picking up the hard way. How to proceed, where to look, how to determine the root cause of the problems that have crept up — all of these are skills that we generally develop over time.

The life cycle of a troubleshooting session usually involves:

- **detection** — noticing that a problem exists
- **identification** — getting a handle on what the problem is
- **analysis** — determining what caused the problem
- **correction** — fixing whatever was wrong
- **prevention** — taking steps to ensure the problem doesn't happen again

A systematic approach to troubleshooting can help to more quickly pinpoint the root cause of a problem that breaks a server or application. **Here are some steps to take and questions to ask yourself:**

1 What just changed?

The most common first reaction to something that stops working is to ask "OK, so what changed?" Looking into recent changes is also the action most likely to pay off if, in fact, some significant change was just made. Look for files, especially configuration files, that might have been modified, applications or packages that were just added, services that were just started, etc.

Don't overlook the fact that many system problems are slow to emerge and looking for something that just changed might not lead you any closer to the cause of whatever problem you're grappling with.

Examples of things that go wrong that are not tied to some change that was just recently made include:

- slowly running out of disk space
- bumping into a configuration flaw that simply never got activated before because certain conditions hadn't yet been met

2 What errors am I seeing?

Pay close attention to any errors that are being displayed on the system console or in your log files. Do those errors point to any particular cause?

Have you seen errors like these before? Do you see any evidence of the same errors in older log files or on other systems? What do online searches tell you? No matter what kind of problem you've run into, you're not likely to be the first sysadmin who has run into them.

3 How is the system or service behaving?

Looking into the symptoms of the problem is also likely to pay off. Is the system or service slow or completely unusable? Maybe only some people cannot log in. Maybe only some functions are not

THINKSTOCK

working. Noticing what works and what doesn't might help you focus on what's wrong.

4 How is this system different than one that is still working?

If you're lucky enough to have redundant systems and have a chance to compare the one that isn't working with one that is, you may be able to identify key differences that can help lead to the cause.

5 What are the likely break points?

Think about how the application or service works and how/where it is likely to have problems. Does it rely on a configuration file? Does it need to communicate with other servers? Is a database involved? Does it write to specific log files? Does it involve multiple processes? Can you easily determine whether all of the required processes are running? If you can, systematically eliminate the potential causes.

6 What troubleshooting tools do I have on hand that might be helpful?

Think about the tools that you have on hand for looking into system problems. Some that might prove useful include:

- **top** — for looking at performance, including some memory, swap space, and load issues
- **df** — for examining disk usage
- **find** — for locating files that have been modified in the last day or so
- **tail -f** — for viewing recent log entries and watching to see if errors are still arriving
- **lsof** — to determine what files a particular process has open
- **ping** — quick network checking
- **ifconfig** — checking network interfaces
- **traceroute** — checking connections to remote systems
- **netstat** — examining network connections
- **nslookup** — checking host resolutions
- **route** — verifying routing tables
- **arp** — checking IP address to MAC address entries in your cache

7 Is anything nasty going on?

Don't rule out the possibility that someone has been messing with your system, although most hackers would prefer to do their work without you noticing anything.

8 What should I NOT do?

Don't confuse symptoms and causes. Whenever you identify a problem, ask yourself why the problem exists.

Be careful not to destroy "evidence" as you work feverishly to get your system back online. Copy log files to another system if you need to recover disk space to get the system back to an operational state. Then you can examine them later to help figure out what caused the problems you're working to resolve. If you need to repair a configuration file, first make a copy of the file (e.g., cp -p config config.save) so that you can more easily look into how and when the file was modified and what you had to do to get things working.

Keep in mind that you might end up making a lot of changes in the process of tracking down your problem. Later on, you might want to think through which of those changes actually resolved the problem.

9 What SHOULD I do?

Record your actions. If you're using PuTTY to connect (or some other tool that allows you to record your system interactions), turn on logging. This will help you when you have to review what happened and how you got past the problem. If you're not out of disk space, you also have the option of using the script command to record your login session (e.g., script troubleshooting.`date %m%d%y`).

If you can't record, keep notes on what you did and what you saw. You might not remember it all later, especially if you're stressed. You might remember the steps, but not the order in which you ran them.

After the problem is resolved, document what happened. You might see it again, and you might need to explain to your boss or your customers what happened and how you're going to prevent it from happening in the future.

Whenever possible, think about how the problem could be avoided in the future. Can you improve your monitoring services so that disk space, memory and network issues, configuration changes, etc. are brought to your attention long before they affect running services?

Wrap up

Good troubleshooting skills can really save the day, and having a plan of attack when a problem arises can play a major role in getting your systems and applications back online and you back home at a decent hour. ◆

Troubleshooting Oracle Connections

When Oracle databases aren't performing, it's wise to check the connections to it. Here's how. **BY SANDRA HENRY-STOCKER**

To comment on this story, visit Network World's Facebook page.

I OFTEN FIND MYSELF needing to verify whether an Oracle database is working. Sometimes a process running on one server extracts data from a database on another server and, if the process fails, testing the connection to the database is a good starting point from which to determine what went wrong. I don't have a lot of Oracle tricks up my sleeve, but a small amount of information about accessing Oracle goes a long way when troubleshooting potential problems.

Checking a Local Database

If the database is on the local system, I will check the process status and expect to see output like that shown below.

```
boson$ ps -ef | grep oracle
    oracle      308       1      0      Oct 11 ?      3:35 ora_dbw0_ORCL
    oracle      338       1      0      Oct 11 ?      0:00 /opt/u01/app/oracle/product/8.1.6/bin
```

```
/tnslsnr LISTENER -inherit
        oracle     306      1     0     Oct 11 ?     0:03 ora_pmon_ORCL
        oracle     310      1     0     Oct 11 ?     1:12 ora_lgwr_ORCL
        oracle     312      1     0     Oct 11 ?     6:31 ora_ckpt_ORCL
        oracle     314      1     0     Oct 11 ?     0:21 ora_smon_ORCL
        oracle     316      1     0     Oct 11 ?     0:01 ora_reco_ORCL
        oracle     318      1     0     Oct 11 ?     0:14 ora_snp0_ORCL
        oracle     320      1     0     Oct 11 ?     0:14 ora_snp1_ORCL
        oracle     322      1     0     Oct 11 ?     0:15 ora_snp2_ORCL
        oracle     324      1     0     Oct 11 ?     0:15 ora_snp3_ORCL
        oracle     7388     1     0     Oct 21 ?     0:00 oracleORCL (DESCRIPTION=(LOCAL=no)

(ADDRESS=(PROTOCOL=BEQ)))
        oracle     908      1     0     Oct 11 ?     0:11 oracleORCL (DESCRIPTION=(LOCAL=no)

(ADDRESS=(PROTOCOL=BEQ)))
        oracle     7414     1     0     Dec 21 ?     0:23 oracleORCL (DESCRIPTION=(LOCAL=no)

(ADDRESS=(PROTOCOL=BEQ)))
```

This output tells me that the Oracle processes are running, including tnslsnr, often referred to as “the listener” and critical for connections to be made to the local database.

I might also check netstat output to see where there is a listen on the port that Oracle uses by default. This is the port that the listener should have opened.

```
boson$ netstat -a | grep 1521
      *.1521              *.*                0       0       0          0   LISTEN
boson.32806         boson.1521           32768   0   32768          0   ESTABLISHED
boson.1521          boson.32806          32768   0   32768          0   ESTABLISHED
localhost.32841     localhost.1521       32768   0   32768          0   ESTABLISHED
localhost.1521      localhost.32841      32768   0   32768          0   ESTABLISHED
boson.1521          10.9.1.13.32854      24820   0    8760          0   ESTABLISHED
boson.1521          10.9.1.13.32861      24820   0    8760          0   ESTABLISHED
```

Using tnsping

Another, potentially more useful command, is tnsping which uses information in Oracle’s tnsnames.ora file to test connectivity. If I type the command “tnsping ORCL”, for example, the command will look for a database with a service name of ORCL and will issue a ping-like request to the configured port and report on the response. **A successful response from tnsping will look something like this:**

```
$ tnsping ORCL
TNS Ping Utility for Solaris: Version 8.1.6.0.0 - Production on 29-DEC-2005
(c) Copyright 1997 Oracle Corporation.  All rights reserved.
Attempting to contact (ADDRESS=(PROTOCOL=TCP)(HOST=boson)(PORT=1521))
OK (11 msec)
```

Notice the "OK" response in the last line. I also get a report on how much time the response took. Eleven seconds is a quick response. Were I checking on a distant system, I wouldn't be surprised if the response took ten or more times as long to get to me.

If Oracle is listening on a port other than 1521, the tnsping command will still work, assuming that the tnsnames.ora file contains the correct information. **A tnsnames.ora entry for a single database will look something like this:**

```
ORCL =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = boson)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = ORCL)
  )
 )
```

Notice how the service name (ORCL) that I used in the example tnsping command and the port on which the Oracle listener is responding (1521) are both configured in this database descriptor.

Using sqlplus

The other command that I like to use to verify connectivity with an Oracle database is sqlplus. With sqlplus (installed along with Oracle), I can both verify that Oracle is responding and I can issue some sqlplus commands to verify that the tables or the content of tables match what I expect to see. **The sqlplus command takes this form to connect to an Oracle database:**

```
sqlplus username/password@SID
```

The SID in this description might be the SID or the SERVICE_NAME, interchangeable in some versions of Oracle. **For example, I might type:**

```
% sqlplus admin/f0xtrawt@ORCL
```

This command specifies the username, password and SERVICE_NAME/SID in a single command. Once connected, I might issue a couple sql commands to verify that the database appears to be working correctly. **I can count the number of user tables with a command like this:**

```
SQL> select count(*) from user_tables;
 COUNT(*)
----------
     88
```

Creating a Script

Because I don't troubleshoot Oracle databases all that often, I prefer to save the commands for connecting to a particular database in a simple script. I might call a script for connecting to the local database "connect2local" and a similar script for connecting to a database on a remote server "connect2rem" or "connect2mars" (if the remote system were named "mars").

- **To facilitate my use of Oracle, I will store the environment variables that I want to use in a profile and source it as needed:**

```
# Oracle Settings
stty istrip
stty erase ^H
export ORACLE_BASE=/opt/oracle
export ORACLE_HOME=$ORACLE_BASE/product/8.1.6
export ORACLE_TERM=vt100
export PATH=$ORACLE_HOME/bin:/usr/ccs/bin:/bin:/usr/bin:/usr/local/bin:/usr/ucb
```

- **The script that I put together will look like this:**

```
#!/bin/bash
#  connect2local: connect to local db using sqlplus

. ~/oracle_profile

# check connectivity to database
tnsping ORCL

sqlplus admin/f0xtrawt@ORCL
```

When I type "connect2local", I will expect to see the OK response and be left at the SQL> prompt from which I can issue queries or type "quit" to exit.

Quick tests to determine whether an Oracle database is responsive go a long way in troubleshooting connection problems. ◆

Linux Commands for Managing, Partitioning, Troubleshooting

Managing Linux disks and the file systems that reside on them is **something of an art** – from initial setup to monitoring performance. **BY SANDRA HENRY-STOCKER**

To comment on this story, visit Network World's Facebook page.

HOW MUCH DO YOU NEED TO KNOW ABOUT DISKS to successfully manage a Linux system? What commands do what? How do you make good decisions about partitioning? What kind of troubleshooting tools are available? What kind of problems might you run into? This article covers a lot of territory – from looking into the basics of a Linux file system to sampling some very useful commands.

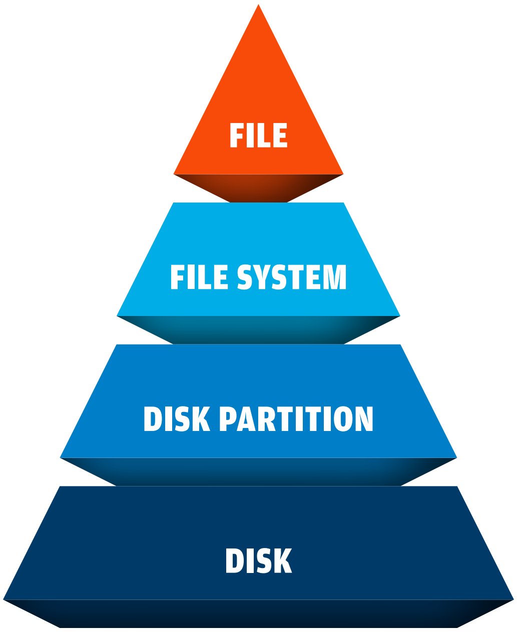
Disk technology

In the beginning days of Unix and later Linux, disks were physically large, but very small in terms of storage capacity. A 300 megabyte disk in the mid-90's was the size of a shoebox. Today, you can get multi-terrabyte disks that are the size of a slice of toast.

Traditionally, files resided within file systems that resided in disk partitions that were themselves simply slices of disks. This organization still dominates today, though servers in large data centers often take on an entirely different structure.

THINKSTOCK

This simplistic view still works for many systems, but these days there are lots of complexities that make disk management harder in some ways and easier in others. A file system might be virtual — no longer residing on a single disk and more complex to manage, but far easier to resize as needed. In fact, the entire system could be virtual. And what we might manage as if it were a single disk could actually be some portion of a very large disk array.



Disk management

Sysadmins generally have to deal with many issues when it comes to managing disks.

These include:

- Partitioning disks
- Creating file systems
- Mounting file systems
- Sharing file systems
- Monitoring free space within file systems
- Backing up (and sometimes restoring) file systems

The reasons to partition a disk include:

- protecting some file systems from running out of space (e.g., you may want the OS partition to be separated from home directories or applications to keep it from being affected if users' files begin to take up an excessive amount of disk space)
- improving performance
- allocating swap space
- facilitating maintenance and backups (e.g., you might be able to unmount /apps if it's not part of / and you might want to back up /home more frequently than /usr)
- more efficient (and targeted) fsck
- maintaining (particularly on test systems) multiple operating systems
- reserving enough disk space for file system expansion
- sharing select file systems with other systems

Partitioning commands

For most Linux servers, partitioning is done before the servers are deployed. On the other hand, you might add disks at some later time or hold back some significant amount of free disk space for future use.

To make changes or verify partitions, enter a command such as **fdisk /dev/sda** to start fdisk interactively and then type m to see a list of the things that you can do with the fdisk command.

```
$ sudo fdisk /dev/sda

Command (m for help): m
Command action
     a    toggle a bootable flag
     b    edit bsd disklabel
     c    toggle the dos compatibility flag
     d    delete a partition
     l    list known partition types
     m    print this menu
     n    add a new partition
     o    create a new empty DOS partition table
     p    print the partition table
     q    quit without saving changes
     s    create a new empty Sun disklabel
     t    change a partition's system id
     u    change display/entry units
     v    verify the partition table
     w    write table to disk and exit
     x    extra functionality (experts only)
```

As you can see on the left, the fdisk command provides a lot of functionality. The partitions that you set up may look something like this configuration in which four partitions have been set up on a single disk – /dev/sda.

```
sda
+-----------------------+----------------------------------+-----------------------------+-------------------+
|      / 40G            |       /home 80G                  |      /apps 70G              |     swap          |
+-----------------------+----------------------------------+-----------------------------+-------------------+
sda1                 sda2                                 sda3                                  sda4
```

Examining disk space and disk partitions

There are a number of excellent commands for examining disk partitions. The df command is one of the most commonly used commands for reporting on disk space usage. With the -h option, the df command displays the measurements in the most "human-friendly" format and that is, in fact, what the "h" is meant to imply. As you can see in the example below, the measurements are displayed in kilobytes, megabytes or gigabytes depending on the sizes rather than all using the same scale.

```
$ df -h
Filesystem       Size     Used     Avail    Use%    Mounted on
udev             969M     4.0K     969M       1%    /dev
tmpfs            196M     1.1M     195M       1%    /run
/dev/sda1         37G     4.5G      31G      13%    /
none             4.0K        0     4.0K       0%    /sys/fs/cgroup
none             5.0M        0     5.0M       0%    /run/lock
none             980M     152K     979M       1%    /run/shm
none             100M      36K     100M       1%    /run/user
/dev/sda3         28G      44M      26G       1%    /apps
```

The pydf command (think "python df" as it's really a python script) also provides a very useful disk usage display showing mount points and cute little illustrations for how full each partition is.

```
$ pydf
Filesystem       Size     Used     Avail    Use%    Mounted on
/dev/sda1         37G    4534M      30G     12.1    [##...........] /
/dev/sda3         27G      44M      26G      0.2    [.............] /apps
```

The parted command displays partition information in a different format:

```
$ sudo parted -1
Model: ATA WDC WD800AAJS-60 (scsi)
Disk /dev/sda: 80.0GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number    Start     End      Size     Type      File system       Flags
 1        1049kB    40.0GB   40.0GB   primary   ext4              boot
 2        40.0GB    50.0GB   10.0GB   primary   linux-swap(v1)
 3        50.0GB    80.0GB   30.0GB   primary   ext4
```

The lsblk (list block devices) command illustrates the relationship between disks and their partitions graphically and also supplies the major and minor device numbers and mount points.

```
$ lsblk
NAME      MAJ:MIN   RM    SIZE   RO   TYPE   MOUNTPOINT
sda         8:0      0   74.5G    0   disk
├─sda1      8:1      0   37.3G    0   part   /
├─sda2      8:2      0    9.3G    0   part   [SWAP]
└─sda3      8:3      0     28G    0   part   /apps
```

The fdisk command reports more details on disk partitions and uses very different numbers. You can also use fdisk to create or delete partitions, list unpartitioned space, change a partition type or verify the partition table.

```
$ sudo fdisk -1

Disk /dev/sda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders, total 156301488 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000f114b

Device      Boot      Start        End      Blocks   Id   System
/dev/sda1    *         2048   78125055    39061504   83   Linux
/dev/sda2          78125056   97656831     9765888   82   Linux swap / Solaris
/dev/sda3          97656832  156301311    29322240   83   Linux
```

The sfdisk command is similar to fdisk, but makes some partition manipulation activities easier to perform.

```
$ sudo sfdisk -1 -uM

Disk /dev/sda: 9729 cylinders, 255 heads, 63 sectors/track
Units = mebibytes of 1048576 bytes, blocks of 1024 bytes, counting from 0

Device      Boot   Start     End     MiB    #blocks   Id   System
/dev/sda1    *         1   38146   38146   39061504   83   Linux
/dev/sda2          38147   47683    9537    9765888   82   Linux swap / Solaris
/dev/sda3          47684   76318   28635   29322240   83   Linux
/dev/sda4              0       -       0          0    0   Empty
```

NOTE: A mebibyte (MiB) = 220 bytes or 1,048,576 bytes.

The cfdisk command can also be used to display or manipulate disk partitions.

```
$ sudo cfdisk

                          cfdisk (util-linux 2.20.1)

                            Disk Drive: /dev/sda
                     Size: 80026361856 bytes, 80.0 GB
             Heads: 255   Sectors per Track: 63   Cylinders: 9729

Name       Flags       Part Type     FS Type          [Label]         Size (MB)
-------------------------------------------------------------------------------
                       Pri/Log       Free Space                           1.05*
sda1       Boot        Primary       ext4                             39998.99*
sda2                   Primary       swap                             10000.27*
sda3                   Primary       ext4                             30025.98*
                       Pri/Log       Free Space                           0.10*

   [ Help ]   [ New ]   [ Print ]   [ Quit ]   [ Units ]   [ Write ]

                    Create new partition from free space
```

Monitoring disk performance

The iostat command can display statistics that illustrate how disks are performing, including how heavily they are being used. It also displays important measurements that show how busy the CPU is and how much of its resources are used for types of work. The system described below is idle more then 95% of the time. More importantly for our focus on disks, the %iowait (CPU waiting on disk IO) is very low. This would not be true if the disk were unusually busy and disk IO were a bottleneck.

```
$ iostat -x 60
Linux 3.13.0-129-generic (stinkbug)   08/31/2017   _x86_64_     (2 CPU)

avg-cpu:          %user          %nice          %system          %iowait          %steal          %idle
                   0.93           1.15           0.35             1.86             0.00            95.73

Device:   rrqm/s   wrqm/s   r/s     w/s    rkB/s    wkB/s    avgrq-sz  avgqu-sz  await  r_await  w_await  svctm  %util
sda       8.37     3.26     13.41   2.79   341.14   191.82   65.79     0.61      37.60  30.40    72.14    2.52   4.08
```

Probably one of the most informative commands for looking at disk health is **smartctl** (part of smartmontools). While the command generates a lot of output, it provides valuable measurements that might help you pinpoint disk problems, particularly once you get used to working with its extensive output.

```
$ sudo smartctl -a /dev/sda1
smartctl 6.2 2013-07-26 r3841 [x86_64-linux-3.13.0-129-generic] (local build)
Copyright (C) 2002-13, Bruce Allen, Christian Franke, www.smartmontools.org


=========================== START OF INFORMATION SECTION ============================


            Model Family:   Western Digital Caviar Blue Serial ATA
            Device Model:   WDC WD800AAJS-60M0A0
           Serial Number:   WD-WMAV37134378
        LU WWN Device Id:   5 0014ee 0015c85ef
        Firmware Version:   02.03E02
           User Capacity:   80,026,361,856 bytes [80.0 GB]
             Sector Size:   512 bytes logical/physical
               Device is:   In smartctl database [for details use: -P show]
          ATA Version is:   ATA8-ACS (minor revision not indicated)
         SATA Version is:   SATA 2.5, 3.0 Gb/s
           Local Time is:   Thu Aug 31 15:30:19 2017 EDT
        SMART support is:   Available - device has SMART capability.
        SMART support is:   Enabled
```

```
========================== START OF READ SMART DATA SECTION ==========================
SMART overall-health self-assessment test result: PASSED

General SMART Values:
Offline data collection status:       (0x82)   Offline data collection activity
                                               was completed without error.
                                               Auto Offline Data Collection: Enabled.
Self-test execution status:           (    0)  The previous self-test routine
                                               completed without error
                                               or no self-test has ever been run.
Total time to complete
offline data collection:              ( 2700)  seconds.
Offline data collection
capabilities:                         (0x5b)   SMART execute Offline immediate.
                                               Auto Offline data collection on/off support.
                                               Suspend Offline collection upon new command.
                                               Offline surface scan supported.
                                               Self-test supported.
                                               No Conveyance Self-test supported.
                                               Selective Self-test supported.
SMART capabilities:                   (0x0003) Saves SMART data before entering
                                               power-saving mode.
                                               Supports SMART auto save timer.
Error logging capability:             (0x01)   Error logging supported.
                                               General Purpose Logging supported.
Short self-test routine
recommended polling time:             (    2)  minutes.
Extended self-test routine
recommended polling time:             (  36)  minutes.
SCT capabilities:                     (0x303f) SCT Status supported.
                                               SCT Error Recovery Control supported.
                                               SCT Feature Control supported.
                                               SCT Data Table supported.
```

```
SMART Attributes Data Structure revision number: 16
Vendor Specific SMART Attributes with Thresholds:
ID#  ATTRIBUTE_NAME           FLAG     VALUE  WORST  THRESH  TYPE      UPDATED  WHEN_FAILED  RAW_VALUE
  1  Raw_Read_Error_Rate      0x002f   200    200    051     Pre-fail  Always   –            0
  3  Spin_Up_Time             0x0027   143    140    021     Pre-fail  Always   –            3841
  4  Start_Stop_Count         0x0032   100    100    000     Old_age   Always   –            178
  5  Reallocated_Sector_Ct    0x0033   200    200    140     Pre-fail  Always   –            0
  7  Seek_Error_Rate          0x002f   100    253    051     Pre-fail  Always   –            0
  9  Power_On_Hours           0x0032   058    058    000     Old_age   Always   –            31203
 10  Spin_Retry_Count         0x0033   100    100    051     Pre-fail  Always   –            0
 11  Calibration_Retry_Count  0x0032   100    100    000     Old_age   Always   –            0
 12  Power_Cycle_Count        0x0032   100    100    000     Old_age   Always   –            175
184  End-to-End_Error         0x0033   100    100    097     Pre-fail  Always   –            0
187  Reported_Uncorrect       0x0032   100    100    000     Old_age   Always   –            0
188  Command_Timeout          0x0032   100    100    000     Old_age   Always   –            0
190  Airflow_Temperature_Cel  0x0022   066    062    040     Old_age   Always   –            34
192  Power-Off_Retract_Count  0x0032   200    200    000     Old_age   Always   –            103
193  Load_Cycle_Count         0x0032   200    200    000     Old_age   Always   –            178
196  Reallocated_Event_Count  0x0032   200    200    000     Old_age   Always   –            0
197  Current_Pending_Sector   0x0032   200    200    000     Old_age   Always   –            0
198  Offline_Uncorrectable    0x0030   200    200    000     Old_age   Offline  –            0
199  UDMA_CRC_Error_Count     0x0032   200    200    000     Old_age   Always   –            0
200  Multi_Zone_Error_Rate    0x0008   200    200    000     Old_age   Offline  –            0

SMART Error Log Version: 1
No Errors Logged

SMART Self-test log structure revision number 1
Num  Test_Description    Status                   Remaining  LifeTime(hours)  LBA_of_first_error
# 1  Short offline       Completed without error  00%        30349            –
# 2  Extended offline    Aborted by host          80%        0                –

SMART Selective self-test log data structure revision number 1
      SPAN      MIN_LBA      MAX_LBA      CURRENT_TEST_STATUS
        1          0            0            Not_testing
        2          0            0            Not_testing
        3          0            0            Not_testing
        4          0            0            Not_testing
        5          0            0            Not_testing
Selective self-test flags (0x0):
After scanning selected spans, do NOT read-scan remainder of disk.
If Selective self-test is pending on power-up, resume after 0 minute delay.
```

There are numerous other commands for examining disks and file systems. Those described here are some of the most useful and informative. Using them periodically has advantages as the easiest way to spot problems is becoming so used to the output of commands such as these that you easily spot the kind of differences that might indicate problems. ◆