Linux has many tools available for troubleshooting some are easy to use, some are more advanced.

I/O Wait is an issue that requires use of some of the more advanced tools as well as an advanced usage of some of the basic tools. The reason I/O Wait is difficult to troubleshoot is due to the fact that by default there are plenty of tools to tell you that your system is I/O bound, but not as many that can narrow the problem to a specific process or processes.

Answering whether or not I/O is causing system slowness

To identify whether I/O is causing system slowness you can use several commands but the easiest is the unix command `top`.

```
# top
top - 14:31:20 up 35 min, 4 users, load average: 2.25, 1.74, 1.68
Tasks: 71 total, 1 running, 70 sleeping, 0 stopped, 0 zombie
Cpu(s): 2.3%us, 1.7%sy, 0.0%ni, 0.0%id, 96.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 245440k total, 241004k used, 4436k free, 496k buffers
Swap: 409596k total, 5436k used, 404160k free, 182812k cached
```

From the CPU(s) line you can see the current percentage of CPU in I/O Wait; The higher the number the more cpu resources are waiting for I/O access.

```
wa -- iowait
 Amount of time the CPU has been waiting for I/O to complete.
```

Finding which disk is being written to

The above top command shows I/O Wait from the system as a whole but it does not tell you what disk is being affected; for this we will use the `iostat` command.

```
$ iostat -x 2 5
avg-cpu: %user %nice %system %iowait %steal %idle
 3.66 0.00 47.64 48.69 0.00 0.00

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 44.50 39.27 117.28 29.32 11220.94 13126.70 332.17 65.77 462.79 9.80 2274.71 7.60 111.41
dm-0 0.00 0.00 83.25 9.95 10515.18 4295.29 317.84 57.01 648.54 16.73 5935.79 11.48 107.02
dm-1 0.00 0.00 57.07 40.84 228.27 163.35 8.00 93.84 979.61 13.94 2329.08 10.93 107.02
```

The iostat command in the example will print a report every 2 seconds for 5 intervals; the `-x` tells iostat to print out an extended report.

The 1st report from iostat will print statistics based on the last time the system was booted; **for this reason in most circumstances the first report from iostat should be ignored.** Every sub-sequential report printed will be based on the time since the previous interval. For example in our command we will print a report 5 times, the 2nd report are disk statistics gathered since the 1st run of the report, the 3rd is based from the 2nd and so on.

In the above example the %utilized for sda is 111.41% this is a good indicator that our problem lies with processes writing to sda. While the test system in my example only has 1 disk this type of information is extremely helpful when the server has multiple disks as this can narrow down the search for which process is utilizing I/O.

Aside from %utilized there is a wealth of information in the output of iostat; items such as read and write requests per millisecond(rrqm/s & wrqm/s), reads and writes per second (r/s & w/s) and plenty more. In our example our program seems to be read and write heavy this information will be helpful when trying to identify the offending process.

Finding the processes that are causing high I/O

iotop

```
# iotop
Total DISK READ: 8.00 M/s | Total DISK WRITE: 20.36 M/s
 TID PRIO USER DISK READ DISK WRITE SWAPIN IO> COMMAND
15758 be/4 root 7.99 M/s 8.01 M/s 0.00 % 61.97 % bonnie++ -n 0 -u 0 -r 239 -s 478 -f -b -d /tmp
```

The simplest method of finding which process is utilizing storage the most is to use the command iotop. After looking at the statistics it is easy to identify bonnie++ as the process causing the most I/O utilization on this machine.

While iotop is a great command and easy to use, it is not installed on all (or the main) Linux distributions by default; and I personally prefer not to rely on commands that are not installed by default. A systems administrator may find themselves on a system where they simply cannot install the non-defualt packages until a scheduled time which may

be far too late depending on the issue.

If iotop is not available the below steps will also allow you to narrow down the offending process/processes.

Process list "state"

The ps command has statistics for memory and cpu but it does not have a statistic for disk I/O. While it may not have a statistic for I/O it does show the processes state which can be used to indicate whether or not a process is waiting for I/O.

The ps state field provides the processes current state; below is a list of states from the man page.

```
PROCESS STATE CODES
 D uninterruptible sleep (usually IO)
 R running or runnable (on run queue)
 S interruptible sleep (waiting for an event to complete)
 T stopped, either by a job control signal or because it is being traced.
 W paging (not valid since the 2.6.xx kernel)
 X dead (should never be seen)
 Z defunct ("zombie") process, terminated but not reaped by its parent.
```

Processes that are waiting for I/O are commonly in an “uninterruptible sleep” state or “D”; given this information we can simply find the processes that are constantly in a wait state.

Example:

```
# for x in `seq 1 1 10`; do ps -eo state,pid,cmd | grep "^D"; echo "----"; sleep 5; done
D 248 [jbd2/dm-0-8]
D 16528 bonnie++ -n 0 -u 0 -r 239 -s 478 -f -b -d /tmp
----
D 22 [kswapd0]
D 16528 bonnie++ -n 0 -u 0 -r 239 -s 478 -f -b -d /tmp
----
D 22 [kswapd0]
D 16528 bonnie++ -n 0 -u 0 -r 239 -s 478 -f -b -d /tmp
----
D 22 [kswapd0]
D 16528 bonnie++ -n 0 -u 0 -r 239 -s 478 -f -b -d /tmp
----
D 16528 bonnie++ -n 0 -u 0 -r 239 -s 478 -f -b -d /tmp
----
```

The above for loop will print the processes in a “D” state every 5 seconds for 10 intervals.

From the output above the bonnie++ process with a pid of 16528 is waiting for I/O more often than any other process. At this point the bonnie++ seems likely to be causing the I/O Wait, but just because the process is in an uninterruptible sleep state does not necessarily prove that it is the cause of I/O wait.

To help confirm our suspicions we can use the `/proc` file system. Within each processes directory there is a file called “io” which holds the same I/O statistics that iotop is utilizing.

```
# cat /proc/16528/io
rchar: 48752567
wchar: 549961789
syscr: 5967
syscw: 67138
read_bytes: 49020928
write_bytes: 549961728
cancelled_write_bytes: 0
```

The `read_bytes` and `write_bytes` are the number of bytes that this specific process has written and read from the storage layer. In this case the bonnie++ process has read 46 MB and written 524 MB to disk. While for some processes this may not be a lot, in our example this is enough write and reads to cause the high i/o wait that this system is seeing.

Finding what files are being written too heavily

The lsof command will show you all of the files open by a specific process or all processes depending on the options provided. From this list one can make an educated guess as to what files are likely being written to often based on the size of the file and the amounts present in the “io” file within /proc.

To narrow down the output we will use the `-p <pid>` options to print only files open by the specific process id.

```
# lsof -p 16528
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bonnie++ 16528 root cwd DIR 252,0 4096 130597 /tmp
<truncated>
bonnie++ 16528 root 8u REG 252,0 501219328 131869 /tmp/Bonnie.16528
bonnie++ 16528 root 9u REG 252,0 501219328 131869 /tmp/Bonnie.16528
bonnie++ 16528 root 10u REG 252,0 501219328 131869 /tmp/Bonnie.16528
bonnie++ 16528 root 11u REG 252,0 501219328 131869 /tmp/Bonnie.16528
bonnie++ 16528 root 12u REG 252,0 501219328 131869 <strong>/tmp/Bonnie.16528</strong>
```

To even further confirm that these files are being written to the heavily we can see if the /tmp filesystem is part of sda.

```
# df /tmp
Filesystem 1K-blocks Used Available Use% Mounted on
/dev/mapper/workstation-root 7667140 2628608 4653920 37% /
```

From the output of df we can determine that /tmp is part of the root logical volume in the workstation volume group.

```
# pvdisplay
 --- Physical volume ---
 PV Name /dev/sda5
 VG Name workstation
 PV Size 7.76 GiB / not usable 2.00 MiB
 Allocatable yes
 PE Size 4.00 MiB
 Total PE 1986
 Free PE 8
 Allocated PE 1978
 PV UUID CLbABb-GcLB-l5z3-TCj3-IOK3-SQ2p-RDPW5S
```

Using `pvdisplay` we can see that the `/dev/sda5` partition part of the sda disk is the partition that the workstation volume group is using and in turn is where `/tmp` exists. Given this information it is safe to say that the large files listed in the `lsof` above are likely the files being read & written to frequently.