Benchmarking Put your setup through its paces with some benchmarks

Hardware: Test

Are your upgrades and tweaks really boosting the performance of your setup? **Bob Moss** can help you find out by using benchmarks.

Our expert

Bob Moss splits his time between coding things and studying for his computer science degree course.

Any time you've ever replaced a hardware component in a PC, or made a software tweak, and not noticed an improvement, you may have found yourself wondering whether the whole exercise was worth it.

This doesn't need to be the case. The free software community has thousands of different tools and tests to analyse things like CPU speed, how fast you can render a movie and your broadband latency – and all with an extremely high level of accuracy and detail.

These details alone don't solve the problem, but benchmarking can. For those who haven't tried it, this involves taking a measurement before and after you make a change. You can then use the two results to compare how your tweak affected things. You could use benchmarking for a broader range of information, but, for our purposes, comparing before and after results is sufficient.

You may prefer to stick to the principles set out in the 'What is a good benchmark?' box opposite to gain a truly objective result, but this tutorial will cover how to execute tests on your hardware and monitor how your system is working so you can perform your own analysis and evaluate which areas of your system could do with a boost.

We'll use hard disk testing as our example. The primary concerns here are the filesystem and the speed at which the platters rotate, since these dictate your read and write speed. Internal buffers can also affect performance, because they can reduce the number of read/write operations that need to be run and thus increase performance.

The first tool we'll use is the infamous *hdparm*. We briefly mentioned this in our 'Power up Linux' feature in **LXF124** (subscribers will find the feature on the *Linux Format* website), and is available to download from most distribution package managers. A little-known feature of *hdparm* is that it can give you a concise guide to how fast your drive is after entering just one line in the terminal:

```
hdparm -Tt /dev/hda
```

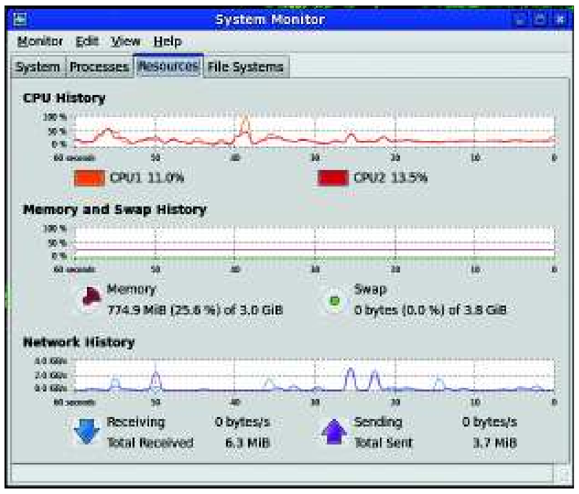
Remember to replace **/dev/hda** with your drive location. The output of the command will show you how much data was read in a fixed time limit, followed by a reflection of that speed per second. The cached-read figure should show a much faster speed than the one below, and the latter is aimed as a direct read access test.

Deeper details

However, if you prefer even more accuracy and detail, then you'll find *Bonnie++* in almost all distribution package managers. It's a very flexible tool indeed, and to get it running you only need to enter the following line in a terminal:

```
bonnie++ /dev/hda
```

Again, substitute **/dev/hda** for the name of your drive. This test will take longer than the previous one, which is to be expected, since *Bonnie++* reads and writes a file 'unintelligently' (with direct disk access), then performs a second test using your hard drive cache. The details you get at the end are much more extensive and, thanks to the

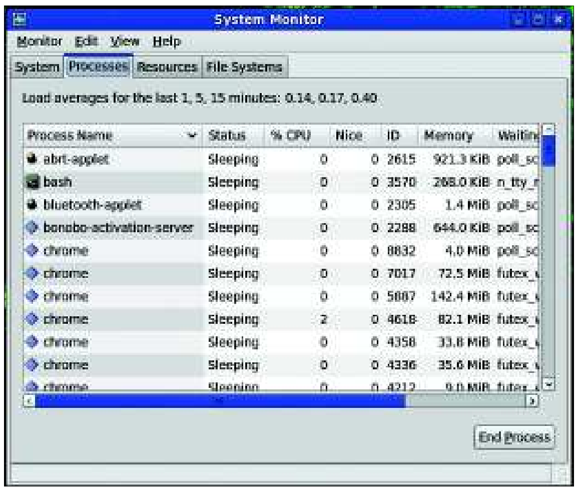


› If terminal instructions don't do it for you, a wiggly line on a graph is a sufficient replacement.

» **If you missed last issue** Call 0870 837 4773 or +44 1858 438795.

performance

LINUX FORMAT On the DVD
» Bonnie++



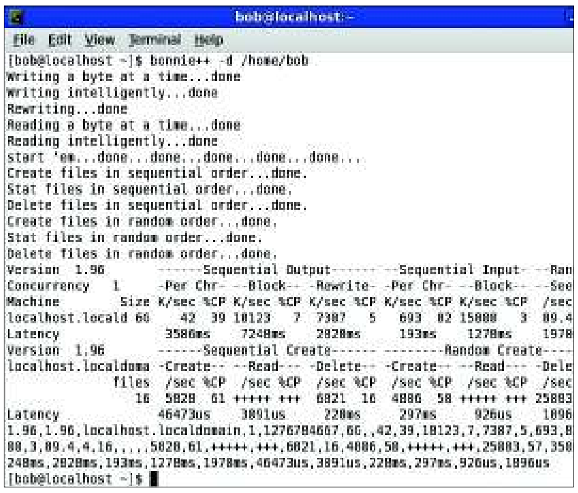
```
System Monitor
Monitor  Edit  View  Help
System | Processes | Resources | File Systems

Load averages for the last 1, 5, 15 minutes: 0.14, 0.17, 0.40

Process Name              Status    % CPU  Nice  ID    Memory     Waiting
abrt-applet               Sleeping      0     0  2615  921.3 KiB  poll_sc
bash                      Sleeping      0     0  3570  268.0 KiB  n_tty_r
bluetooth-applet          Sleeping      0     0  2305    1.4 MiB  poll_sc
bonobo-activation-server  Sleeping      0     0  2288  644.0 KiB  poll_sc
chrome                    Sleeping      0     0  8832    4.0 MiB  poll_sc
chrome                    Sleeping      0     0  7017   72.5 MiB  futex_
chrome                    Sleeping      0     0  5887  142.4 MiB  futex_
chrome                    Sleeping      2     0  4618   82.1 MiB  futex_
chrome                    Sleeping      0     0  4358   33.8 MiB  futex_
chrome                    Sleeping      0     0  4336   35.6 MiB  futex_
chrome                    Sleeping      0     0  4212    9.0 MiB  futex_

                                                        End Process
```

› **Like the *Microsoft Task Manager*, you can view running processes and kill them from here.**



```
bob@localhost:~
File  Edit  View  Terminal  Help
[bob@localhost ~]$ bonnie++ -d /home/bob
Writing a byte at a time...done
Writing intelligently...done
Rewriting...done
Reading a byte at a time...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Create files in sequential order...done.
Stat files in sequential order...done.
Delete files in sequential order...done.
Create files in random order...done.
Stat files in random order...done.
Delete files in random order...done.
Version  1.96       ------Sequential Output------ --Sequential Input- --Ran
Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --See
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec
localhost.locald 6G    42  39 10123   7  7387   5   693  82 15888   3  89.4
Latency              3586ms    7248ms    2828ms     193ms    1278ms    1978
Version  1.96       ------Sequential Create------ --------Random Create----
localhost.localdoma -Create-- --Read--- -Delete-- -Create-- --Read--- -Dele
              files  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec %CP  /sec
                 16  5828  61 +++++ +++  6821  16  4886  58 +++++ +++ 25883
Latency             46473us    3891us     228ms     297ms     926us    1896
1.96,1.96,localhost.localdomain,1,1276784667,6G,,42,39,10123,7,7387,5,693,8
88,3,89.4,4,16,,,,,5828,61,+++++,+++,6821,16,4886,58,+++++,+++,25883,57,358
248ms,2828ms,193ms,1278ms,1978ms,46473us,3891us,228ms,297ms,926us,1896us
[bob@localhost ~]$ 
```

› ***Bonnie++* will put your filesystem through its paces and provide a detailed analysis of the results.**

workout it gives your drive, includes latency figures or the time taken for file reads, creation and deletions completed in a variety of ways.

However, *Bonnie++* isn't just restricted to monitoring the entire drive. You can set the number of times the test is run, the size of files used in the test and the number of them used. Just type **bonnie++** into your terminal to see the vast number of options *Bonnie++* offers to put your hard drive through its paces.

Ferocious Pheronix

If testing your hard drive isn't enough then you could try out *Pheronix Test Suite*. It's available by default in most package managers, but there is a caveat. A GUI should sit on top and run from your Gnome menu, but the library it relies on hadn't been updated to work with modern distributions at the time of writing. This is disappointing, but the command line version of *Pheronix Test Suite* is still useful to us.

To list all the possible tests you can run, try:

```
phoronix-test-suite list-tests
```

and simply replace **list-tests** with **list-suites** to see various options to run every test in a given scenario. Our example will involve having a look at testing audio encode speed, for which we need the following command:

```
phoronix-test-suite run audio-encoding
```

It will take a little time to install the tests on the first run, but once it's complete you should see results for how fast a generic WAV file has been encoded to a number of different container formats.

Finally, you can use this package to find out detailed system information by using:

```
phoronix-test-suite system-info
```

This tutorial is only the tip of the iceberg when it comes to measuring your current system's performance and identifying areas for potential improvement. There's a huge number of free tools and utilities available out there, so it's over to you to experiment and tweak your own setup for maximum performance. LXF

Quick tip

Benchmark tests aren't designed to take all day! If they last longer than 30 minutes, consider adjusting the test to something better suited to your hardware setup.

What is a good benchmark?

Let's say someone tells you, "An Intel Core 2 Solo processor compiles *Firefox* from source 10% faster." There are several issues with this. Firstly, we don't know which processor this is being compared to and which version of the source package is being used for the test. Benchmark tests should be easily repeatable and you should have extensive details about the before and after conditions to compare results.

Next, there are certain things that aren't really comparable in an objective fashion. When you're benchmarking for performance, you should be simply tweaking one thing on nearly identical systems – for instance, finding out whether profiling your kernel actually decreases boot time or not. Checking to see whether Linux boots faster than Windows is a fun test to try, but it isn't really the aim of benchmarking.

Finally, stick to tools for the platform you're currently using. As per the above point, any benchmark results you got from Windows tools can't be compared against benchmark results you get in Linux, because the systems work differently. Using different tools means any difference between your before and after results could be down to the benchmarking algorithms rather than an actual boost in performance.