
Reduce your Linux memory footprint

A few tweaks can boost performance

Skill Level: Intermediate

[Martyn Honeyford \(martynh@uk.ibm.com\)](mailto:martynh@uk.ibm.com)
Software Engineer
IBM

31 Jan 2007

A lack of physical memory can severely hamper Linux® performance. In this article, learn how to accurately measure the amount of memory your Linux system uses. You also get practical advice on reducing your memory requirements using an Ubuntu system as an example.

A much-touted benefit of Linux is that it is more efficient than Microsoft® Windows®, and will therefore perform better on less than cutting-edge hardware. This performance makes Linux a very attractive upgrade for the many people who have old Windows 98-era boxes still sitting around that are no longer being treated to the latest and greatest software (particularly security patches).

The truth of the matter, however, is that while the Linux kernel can still be configured to be reasonably small and efficient, as new computers have increased in power, many Linux desktop environments (such as KDE and GNOME) have added lots of features. Consequently, the default install of most distributions offer a less than stellar level of performance when installed on older hardware. The same is true of many modern applications also -- Web browsers such as Firefox and office suites such as OpenOffice are fully featured, but trying to run them on a machine with 128MB of RAM can be a painful experience!

So what is the answer? Throw away all of your old hardware and upgrade? Install a Linux distribution from circa 1995? (If you decide to go that route, I recall having a good experience with Linux-FT.)

Never fear: as those in the Linux community have known for years, a great strength (some would say *the* great strength) of the Linux kernel and Linux distributions in general lies in their ability to be customized. This article delves into how you can tailor your Linux systems for better performance on modest hardware.

Thanks for the memory

In most cases, the single most important factor in desktop operating system performance is the amount of system memory available. It's all well and good to have a fast processor, but if there isn't enough physical memory to keep that processor utilized, the system will spend all its time shuffling data between physical memory and swap space (a condition known as *thrashing*) and the CPU will spend most of its time idle. For older systems, therefore, adding extra memory is usually the easiest way to improve performance. There are a number of reasons why this may not be possible, however, ranging from a lack of free slots, to a scarcity of affordable RAM for some systems (particularly laptops or RAMBUS-based systems), to an understandable reluctance to spend more money on an aging system.

If you are unable or unwilling to upgrade your RAM, the next best thing is to reduce the RAM requirements on your system. This article shows you five simple steps to memory nirvana for your Linux machine.

Step 1: Choose the right desktop environment

Your single most important choice is the Linux distribution and desktop environment (DE) you're going to install. While these are two distinct choices, the choice of distribution can affect the choice of DE. There is nothing stopping you from installing, say, Fluxbox on Ubuntu; however, you'll find that life is a lot easier if you simply use the default DE that comes with your distribution.

In the scenario in this article, the goal is to find a simple desktop-oriented distribution that is easy for new users. I have started off with Ubuntu 6.10, which comes with GNOME 2.16.

For my base system, I have chosen an old machine with an 800MHz processor and 256MB of RAM. I will execute each of my tests twice, once booting as normal with the full 256MB of RAM and once with `mem=128M` appended to the kernel line, which forces the kernel to only recognize 128MB of the physical memory. This effectively allows me to examine both a 256MB machine and a 128MB machine without actually using a second physical machine (or repeatedly physically removing the RAM chips from my machine and then putting them back in). This kernel-line option provides a close enough approximation of how a 128MB machine will behave; note, however that if you really have only 128MB, you may experience additional complications -- for instance, with Ubuntu you must use an install disk different from the common one for machines with less than 192MB of RAM.

To get a base level of memory usage, I booted the system, logged in to the desktop, and started a terminal (in the rest of this article, I'll refer to this setup as my *base level*); then I checked the amount of free memory using the `free` command, with the results as shown in Listing 1.

Listing 1. Base level for Ubuntu on a 256MB machine

```

ubuntu # free
        total      used      free      shared    buffers
cached
Mem:    255988    231704    24284         0     6432
139292
-/+ buffers/cache:    85980    170008
Swap:   746980         0    746980

```

The first line indicates that, out of 256MB of RAM, 231MB is "in use." The next line shows us that while 231MB is being used, only 86MB of this is actually being used by applications; the rest is being used for buffers and cache.

The most important part of this listing for assessing performance is the `Swap` line; this shows us that we aren't currently using any swap, which suggests that we don't really have any memory problems at the moment. The system is able to fit everything into physical memory without having to resort to slow, disk-based swap space.

Next, to get an idea of what the system would look like under normal day-to-day use, I fired up a Web browser (Firefox 2.0) and pointed it to developerWorks, connected an instant messaging client (Gaim) to MSN, and used the file manager to browse to a folder and open a reasonably sized Microsoft Word-formatted document in OpenOffice. (In the rest of this article, I'll refer to this setup as my *light usage level*.)

Once everything had loaded, the output of the `free` command looked like Listing 2.

Listing 2. Light usage level for Ubuntu on a 256MB machine

```

ubuntu # free
        total      used      free      shared    buffers
cached
Mem:    255988    252196    3792         0     21276
87500
-/+ buffers/cache:    143420    112568
Swap:   746980    18676    728304

```

You can see that things are getting a little leaner in the memory department. Now applications have used 143MB of physical memory, with the rest of being used for buffers; in addition, the system now needs to make use of 18MB of swap. The system generally seemed quite usable under these light office task conditions, but there isn't much head room, and I wouldn't like to start doing anything more demanding, such as editing large digital photos or video, as the system would soon begin to crawl.

To get an idea of how the system would perform with only 128MB, I then rebooted and added `mem=128M` to the kernel line, as I described above. At the same base level state illustrated in Listing 1, with 128MB of RAM I got the results shown in Listing 3.

Listing 3. Base level for Ubuntu on a 128MB machine

```

ubuntu # free
        total      used      free      shared    buffers
cached

```

```

Mem:          126100    121464    4636        0    1636
37000
-/+ buffers/cache:    82828    43272
Swap:        746980    17924    729056

```

You can see now that with 128MB, I am already eating into swap, and I haven't actually started doing anything yet.

Starting my simple set of applications produced the results shown in Listing 4.

Listing 4. Light usage for Ubuntu on a 128MB machine

```

ubuntu # free
      total      used      free      shared      buffers
cached
Mem:    126100    123608    2492        0        392
51208
-/+ buffers/cache:    72008    54092
Swap:   746980    98452    648528

```

As you can probably predict from these numbers, the machine is now a lot less responsive under normal use -- it is still just about usable for these simple tasks, but the disk is being accessed very frequently and I certainly wouldn't like to use it as my main machine. You can see that the total amount of application memory required is approximately 170MB, but only 72MB will fit into physical memory, and so 98MB is relegated to swap. This helps explain the lack of system responsiveness!

For my next set of tests, I decided to use Xubuntu, a distro from a Ubuntu-associated project. This distribution is quite similar to Ubuntu, but uses the Xfce 4.4 Beta 2 DE instead of GNOME. Unlike the more popular GNOME and KDE projects, which concentrate on maximum functionality, Xfce is designed to be lightweight, so it should hopefully be a better match for aging hardware. We'll perform the same tests with this distribution that we performed with Ubuntu.

You can see in Listing 5 that the base DE uses approximately 25MB less application memory, and also eats significantly less into buffers and cache (which may imply that there is less file activity) than Ubuntu.

Listing 5. Base level for Xubuntu on a 256MB machine

```

xubuntu # free
      total      used      free      shared      buffers
cached
Mem:    255988    170964    85024        0        6004
104700
-/+ buffers/cache:    60260    195728
Swap:   746980        0    746980

```

In Listing 6, we start up our suite of test applications again (Web browser, IM client, and word processor). You can see that the memory use appears to be around 20MB less than Ubuntu required for the same set of apps (126 in physical memory and 17 in swap for a total of 143 versus 143 + 18 for a total of 161).

Listing 6. Light usage level for Xubuntu on a 256MB machine

```
xubuntu # free
          total      used      free      shared      buffers
cached
Mem:      255988      252180      3808          0          1972
124008
-/+ buffers/cache:      126200      129788
Swap:     746980      16956      730024
```

Listing 7 illustrates base level usage with only 128MB of RAM. My memory-limited system is much happier this time around, with no swap being used yet.

Listing 7. Base level for Xubuntu on a 128MB machine

```
xubuntu # free
          total      used      free      shared      buffers
cached
Mem:      126100      123228      2872          0          4252
60484
-/+ buffers/cache:      58492      67608
Swap:     746980          0      746980
```

In Listing 8, we start the apps up again. The system is still much better off here than with Ubuntu, but it is still using significant amounts of swap, and the machine is still a bit sluggish (albeit less so than under Ubuntu).

Listing 8. Light usage level for Xubuntu on a 128MB machine

```
xubuntu # free
          total      used      free      shared      buffers
cached
Mem:      126100      123980      2120          0          468
56276
-/+ buffers/cache:      67236      58864
Swap:     746980      64516      682464
```

As you can see from these numbers, Xubuntu generally used less memory across the board; so if you only have 128MB (or less) to work with, it is probably a good choice for your system.

One great thing about Linux distributions is that they generally don't cost anything, so it's easy to download several and try them out for a while to see which you prefer and how they all perform on your hardware. If you have very limited hardware, you may want to look at a distribution like Damn Small Linux, which claims to run on systems with specs as low as a 486DX processor with 16MB of RAM.

As I have a little more breathing room in my 256MB system, and am generally a KDE kind of guy, I have gone with another Ubuntu derivative, called Kubuntu, which is KDE based and seems to sit somewhere in between Xubuntu and Ubuntu in terms of memory use. For reference, Listing 9 illustrates the base usage for Kubuntu.

Listing 9. Base level for Kubuntu on a 256MB machine

```
kubuntu # free
          total      used      free      shared      buffers
cached
Mem:      255988      244736      11252          0          7612
```

```

160008
-/+ buffers/cache:      77116      178872
Swap:      746980          0      746980

```

Step 2: Choose appropriate apps

Once you have settled on a distribution, the next thing you can streamline is the set of applications you are going to use. The memory requirements of different applications can vary greatly; sometimes you face a trade-off between size and functionality, but in other cases even functionally equivalent applications can have vastly different memory requirements.

In order to measure memory usage for the purpose of this article, I will use the tool `exmap`. This tool is more accurate than `ps` or `top` because it takes into account shared libraries in use by multiple applications. For example, if two applications are using the same shared library, which takes up 1MB of memory, `ps` will show both apps using 1MB of extra memory, whereas `exmap` more correctly shows each app using 500 KB. This greater accuracy is particularly important for assessing desktop environments such as KDE and GNOME, which make very heavy use of libraries shared between applications.

For each application I discuss in the following sections, I measured the *resident* and *effective resident* values produced by `exmap`. The resident value represents the total amount of physical memory in use by the process, including shared libraries that are in use by other processes, and is typical of the sort of value you would expect from `ps` or `top`. The effective resident value is the value with shared library allocations shared evenly among all the processes that are using them; this is a much more accurate view of the system memory being consumed by a given process.

Note also that, when determining the total memory footprint of a process, you should also take into account the *mapped* and *effective mapped* values, which are parts of the process that have been placed in swap. The mapped values are analogous to the resident values, but for pages in swap rather than in physical memory. So for a view that doesn't compensate for shared libraries, you would add the resident and mapped values together; for a view compensating for shared libraries, you would add the effective resident and effective mapped values together. I did *not* record these values in the ensuing tables because, for all of my tests, none of the processes were in swap, and so the output of the `exmap` command had a value of zero for those columns.

Web browsers

For each browser in Table 1, I launched the application, opened the developerWorks homepage, and waited for it to fully render. The results are shown in Table 1.

Table 1. Comparison of Web browser memory usage

Application	Effective resident memory (KB)	Resident memory (KB)
Firefox	27708	35068

Opera	20477	27816
Konqueror	13479	29748
Dillo	2776	6888
Lynx	1101	1540

From this table, you can see that there is a very large range in memory usage, with the most memory-hungry browser (Firefox) using approximately 27 times more memory than the most frugal (Lynx). This isn't really a fair comparison, as Lynx is not really functionally equivalent (it does not even display graphics, for instance), but it does show that, depending upon your requirements, you can vastly reduce your memory usage. Even among the first three browsers listed in Table 1, which are more or less functionally equivalent to one another, Opera uses approximately two-thirds the amount of memory of Firefox, and Konqueror uses less than half of Firefox's memory.

For modest use requirements, Dillo may provide a good compromise between a fully featured browser and the austerity of Lynx. Dillo provides a GUI; it is fairly limited in its default state, however, and doesn't even support SSL without an additional plugin!

Note also that when we compare the shared memory use, Konqueror performs much better than Firefox, using around 14MB less memory; if we look at the total use, however, while Konqueror still does better than Firefox, it doesn't do so by such a wide margin -- only by 5MB or so. This is because Konqueror makes heavy use of shared KDE libraries that are loaded into lots of applications, as I am using the KDE desktop. If I were not using any other KDE apps, however, Opera would be a better choice than KDE. We'll discuss this issue in more detail later.

Word processors

To test word processors, I loaded the same Microsoft Word-formatted document I used in my first test into the word processors listed in Table 2.

Table 2. Comparison of word processor memory usage

Application	Effective resident memory (KB)	Resident memory (KB)
OpenOffice Writer	70114	81960
AbiWord	58029	65224
KWord (from KOffice)	46512	60096

You can clearly see from these numbers that OpenOffice Writer uses much more memory than KWord or AbiWord. After OpenOffice, KWord did the best job of rendering the Microsoft Word-formatted document correctly. While AbiWord was able to open the document successfully, there were a few problems rendering it correctly, so if interoperability with Microsoft Office is important to you, you may be better off sticking to OpenOffice.

Instant messaging clients

To test instant messaging, I logged into my MSN Messenger account using the IM clients outlined in Table 3.

Table 3. Comparison of IM client memory usage

Application	Effective resident memory (KB)	Resident memory (KB)
aMSN	18455	20344
Gaim	13456	21464
Kopete	10988	24176
KMess	7154	19660

Here, Kmess came out best for me and was a reasonable choice, as I was only interested in connecting to MSN. If you need to use other services, then Kopete looks like the best choice. Note, however, that the application's memory usage may increase if you're using a different IM protocols; also, Kmess is an integrated KDE app, so if you aren't using KDE, then Gaim may be a better choice for you.

Lather, rinse, repeat

Now that you know how to analyze the memory use of your applications, you can simply repeat the process for all the application types you're interested in, trying out the various options available until you find the choice with the lowest memory requirements that still meets your needs functionally.

As you will have noticed in the Web-browsing section above, it is often the case that the best memory savings come from using an app that is tightly integrated with your desktop environment. This is because such apps make heavy use of shared libraries that are embedded into the DE and are most likely already loaded. For instance, Konqueror is the file manager for KDE as well as a Web browser; hence, it uses much less memory than Firefox when run on a KDE system, because much of its functionality is already loaded by other apps. Similarly, if you then wanted to use an RSS aggregator, Akregator may be a good choice, as it will most likely use those same libraries again.

Thus, if you are concerned about memory usage, it is important that you perform these tests on your *own* system, because it is generally hard to get a sense of which application will use less memory on your system by looking at other people's benchmarks.

This fact may have implications for your choice of DE. If you really want to use Konqueror, for instance, it is probably most efficient to use KDE as your DE. Similarly, if you are a GNOME user, you may want to think twice before you use that simple little KDE app you've had your eye on, as it is likely to load a whole host of libraries that only it will make use of.

Step 3: Remove unwanted services and settings

Once you have chosen a distribution, desktop environment, and selection of apps, what else can you do to further reduce your memory use? To answer this question, you will need to dig down a little deeper and configure your system. Armed with `exmap`, you can analyze what is running on the system and try to remove the things you don't need and configure the things you do.

A good place to begin is with the services that are started automatically when the system boots -- though you need to be careful here so as to not remove anything that is necessary for your system to run. You will need to do some research into what is required by your particular distribution, and on how services are configured, as this will vary by distribution. Some distributions are worse than others, starting by default lots of unneeded services such as Web servers and so on that eat up memory.

Beyond system services, you may also wish to look at how the DE is configured, as it may well be starting services which are not needed.

My Kubuntu box didn't seem to be starting too many unneeded services, but a quick look at the process list did show a few obvious things, which I was able to remove:

- **HPLIP** (4.4MB): Services for HP printers and scanners. Unneeded as no such devices are connected to this computer.
- **cupsd** (1.1MB): A printer daemon. Not needed as no printer is attached to this computer.
- **kbluetoothd** (3.2MB): The KDE Bluetooth daemon. Not needed because this computer doesn't have Bluetooth connectivity.
- **klipper** (1.7MB): The KDE clipboard tool. I don't care for this particular tool, so I disabled it.
- **KMix** (4.1MB): The KDE audio mixer. Doesn't need to be running all the time, as I adjust volume via external speakers.

Just five minutes of configuration yielded memory savings of around 14MB -- not bad from a starting point of around 77MB!

It is worth digging around in the settings of your DE as well as the larger apps you use, as some may have settings that will affect the amount of memory they use. For instance, by reducing the number of virtual desktops, you have may save some memory, particularly if you are using large bitmaps as backdrops. Turning off some of the fancy eye-candy effects may help too.

Step 4: Have appropriate expectations

When running older hardware, it is important to be appreciative of the limitations of

the machine and work appropriately. For instance, if you want to edit a collection of photos, don't open them all at the same time. This will only eat up memory unnecessarily. You would have a much easier time of it opening them one by one and closing them afterwards. Similarly, if you are trying to capture and edit some video, consider capturing individual scenes rather than the whole lot at once; and if you are creating a large document that will contain illustrations, don't add them in until you have finished with the text.

Step 5: Optimize your system

The final step is to look at the deep guts of your system and see if you can shave any memory off anywhere. There are lots of opportunities, but the law of diminishing returns will start to apply, and for most people the amount of pain and work will not be worth it. Still, here are some of the things that you might consider:

- Recompile the kernel with only the specific drivers for your hardware. Most mainstream distributions are tailored for a wide range of hardware, so they often include support for vast swathes of hardware that you aren't using. This may yield some benefit, though it is likely that most of the hardware support is in the form of modules that won't be loaded if they are not required.
- Recompiling certain apps or libraries so that they are optimized for size and targeted to the specific CPU you are using can yield some memory gains. The Gentoo distribution is perfect for this, as you can easily recompile some or all of your system with the precise compilation flags you choose. Unfortunately, this is likely to be quite a lengthy process on an old machine.
- Recompiling apps and libraries with specific features removed can also reduce their memory requirement. Gentoo is again a good idea here because it includes the concept of `USE` flags, which allow you to easily build your system with application features disabled. The size of applications can be significantly reduced in this way -- apps are generally released with support for a large number of file formats, codecs, and so on, which often pull in libraries to provide the actual support. If you know that you never want to read, say, JPEG files, you can specify this in Gentoo (via `USE=" -jpeg "`), and any applications that deal with graphics will be compiled without JPEG support and will most likely have a smaller memory overhead.
- Recent 2.6 kernels have a *swappiness* parameter that can be tuned at runtime. This parameter determines the likelihood that applications will be moved to swap instead of the cache or buffers being reduced -- we saw in our early tests that it is common for applications to be swapped out while large chunks of physical memory are still reserved for caches. By

reducing the tendency to swap out, the cache will be reduced and more apps are likely to stay in memory. Whether or not this move will actually make your machine run faster will very much depend on what apps you are running, however. It may make the machine seem more responsive if you are constantly swapping between applications, because they are more likely to be resident in memory; however, if you have a task that reads and writes to disk a lot, then it is likely to be slower. In general, reducing swappiness will make your system more responsive for interactive apps but will slow down overall throughput.

Conclusion



The ideas in this article may help you breathe life (and some additional security) into your old machines and make better use of Linux on aging hardware. The measurements show that an 800-MHz/256-MB machine can house a perfectly usable Linux desktop for light office and home tasks such as e-mail, browsing the Web, word processing, and so on. With a little tweaking and experimentation, even a 128MB machine can serve as a reasonably comfortable desktop computer.

While the main focus of this article has been to achieve a functional desktop on reasonably limited hardware, you can also apply the same principles to most sectors of Linux use. No matter how much memory your latest super machine has, you will soon find new apps to fill it with. By applying these techniques, you may be able to squeeze that little extra performance out of your overloaded servers or gain some insight into the memory use within your own applications.

Resources

Learn

- Read about [memory usage](#) in a pair of useful articles from Lubos Lunak.
- In the [developerWorks Linux zone](#), find more resources for Linux developers.
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Learn more about the [Exmap](#) memory analysis tool.
- Learn more about the distributions discussed in this article:
 - [Ubuntu](#)
 - [Kubuntu](#)
 - [Xubuntu](#)
 - [Damn Small Linux](#)
- [Order the SEK for Linux](#), a two-DVD set containing the latest IBM trial software for Linux from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.
- With [IBM trial software](#), available for download directly from developerWorks, build your next development project on Linux.

Discuss

- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the author

Martyn Honeyford

Martyn Honeyford graduated from Nottingham University with a BSc in Computer Science in 1996. He has worked as a software engineer at IBM UK Labs in Hursley, England, ever since. He has given all hope of learning to play the guitar and instead spends his rare moments of free time playing video games until his hands stop working. Until his international snowboarding career takes off, his day job is as a developer in the WebSphere Message Broker development team.