

Home > Linux

What Is Swappiness on Linux? (and How to Change It)

Swappiness in Linux has nothing to do with how much RAM is used before swapping starts. We tell you what it really is.

BY DAVE MCKAY PUBLISHED DEC 12, 2019

Link copied to clipboard

Re Geek. When you make a purchase using links on our site, we may earn an affiliate

Quick Links

Busting Myths About Swappiness

Your RAM is Split Into Zones

The PAGESIZE Value

Zones Are Attached to Nodes

File Pages and Anonymous Pages

Swappiness

The Golden Ratio

When Does Swap Actually Cut In?

What Should Swappiness Be Set To?

How to Set the Linux Swappiness Value

Memory Management is Complex


The Linux swappiness value has nothing to do with how much RAM is used before swapping starts. That's a widely reported and widely believed mistake. We explain what it really is.

Busting Myths About Swappiness

Swapping is a technique where data in Random Access Memory (RAM) is written to a special location on your hard disk---either a swap partition or a swap file---to free up RAM.


ANDROID

IPHONE




What Is "Watch Only" Mode on a Galaxy Watch? (and How to Use It)

1 day ago




How to Uninstall Android Apps From Your Phone or Tablet

2 days ago



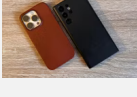
Google Podcasts Is Shutting Down in 2024

2 days ago



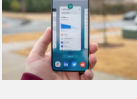
How to Set Up Android's Guest Mode

3 days ago



I Bought a Leather Phone Case and I'm Never Going Back

3 days ago



How to See How Much RAM Your Android Phone Has

3 days ago

See More

TRENDING NOW

https://archive.ph/CtH3f

1/10

Linux has a setting called the swappiness value. There's a lot of confusion about what this setting controls. The most common incorrect description of swappiness is that it sets a threshold for RAM usage, and when the amount of used RAM hits that threshold, swapping starts.

This is a misconception that has been repeated so often that it is now received wisdom. If (almost) everyone else tells you that's exactly how swappiness works, why should you believe us when we say it isn't?

Simple. We're going to prove it.

## Your RAM is Split Into Zones

Linux doesn't think of your RAM as one big homogenous pool of memory. It considers it to be divided into a number of different regions called zones. Which zones are present on your computer depends on whether it is [32-bit](#) or [64-bit](#). Here's a simplified description of the possible zones on an [x86 architecture computer](#).

- **Direct Memory Access (DMA):** This is the low 16 MB of memory. The zone gets its name because, a long time ago, there were computers that could only do direct memory access into this area of physical memory.
- **Direct Memory Access 32:** Despite its name, Direct Memory Access 32 (DMA32) is a zone only found in 64-bit Linux. It's the low 4 GB of memory. Linux running on 32-bit computers can only do DMA to this amount of RAM (unless they are using the [physical address extension](#) (PAE) kernel), which is how the zone got its name. Although, on 32-bit computers, it is called HighMem.
- **Normal:** On 64-bit computers, normal memory is all of the RAM above 4GB (roughly). On 32-bit machines, it is RAM between 16 MB and 896 MB.
- **HighMem:** This only exists on 32-bit Linux computers. It is all RAM above 896 MB, including RAM above 4 GB on sufficiently large machines.

## The PAGESIZE Value

RAM is allocated in pages, which are of a fixed size. That size is determined by the kernel at boot time by detecting the architecture of the computer. Typically the page size on a Linux computer is 4 Kbytes.

You can see your page size [using the](#)

```
getconf
```

[command:](#)

```
getconf PAGESIZE
```

```
dave@howtogeek:~$ getconf PAGESIZE
4096
dave@howtogeek:~$
```



**Ugreen Nexode 100W 2-in-1 GaN Charger Review: Have Power, Will Travel**



**Apple Watch Ultra 2 Review: The Second Ultra Makes Ripples Not Waves**



**How to Add Text to Your iPhone Lock Screen**

## Zones Are Attached to Nodes

Zones are attached to nodes. Nodes are associated with a [Central Processing Unit \(CPU\)](#). The kernel will try to allocate memory for a process running on a CPU from the node associated with that CPU.

The concept of nodes being tied to CPUs allows mixed memory types to be installed in specialist multi-CPU computers, using the [Non-Uniform Memory Access](#) architecture.

That's all very high-end. The average Linux computer will have a single node, called node zero. All zones will belong to that node. To see the nodes and zones in your computer, look inside the `/proc/buddyinfo` file. We'll use `less` to do so:

```
less /proc/buddyinfo
```

```
dave@howtogeek:~$ less /proc/buddyinfo
```

This is the output from the 64-bit computer this article was researched on:

```
Node 0, zone DMA 1 1 1 0 2 1 1 0 1 1 3
```

```
Node 0, zone DMA32 2 67 58 19 8 3 3 1 1 17
```

There is a single node, node zero. This computer only has 2 GB of RAM, so there is no "Normal" zone. There are only two zones, DMA and DMA32.

Each column represents the number of available pages of a certain size. For example, for the DMA32 zone, reading from the left:

- **2:** There are 2 of  $2^{(0 \times \text{PAGE\_SIZE})}$  chunks of memory.
- **67:** There are 67 of  $2^{(1 \times \text{PAGE\_SIZE})}$  chunks of memory.
- **58:** There are 58 of  $2^{(2 \times \text{PAGE\_SIZE})}$  chunks of memory available.
- And so on, all the way up to...
- **17:** There are 17 of  $2^{(512 \times \text{PAGE\_SIZE})}$  chunks.

But really, the only reason we're looking at this information is to see the relationship between nodes and zones.

## File Pages and Anonymous Pages

Memory mapping uses sets of [page table entries](#) to record which memory pages are used, and for what.

Memory mappings can be:

- **File backed:** File backed mappings contain data that has been read from a file. It can be any kind of file. The important thing to note is that if the system freed this

memory and needed to obtain that data again, it can be read from the file once more. But, if the data has been changed in memory, those changes will need to be written to the file on the hard drive before the memory can be freed. If that didn't happen, the changes would be lost.

- **Anonymous:** Anonymous memory is a memory mapping with no file or device backing it. These pages may contain memory requested on-the-fly by programs to hold data, or for such things as the [stack](#) and the [heap](#). Because there is no file behind this type of data, a special place must be set aside for the storage of anonymous data. That place is the swap partition or swap file. Anonymous data is written to swap before anonymous pages are freed.
- **Device backed:** Devices are addressed through [block device files that can be treated as though they were files](#). Data can be read from them and written to them. A device backed memory mapping has data from a device stored in it.
- **Shared:** Multiple [page table entries](#) can map to the same page of RAM. Accessing the memory locations through any of the mappings will show the same data. Different processes can communicate with one another in a very efficient way by changing the data in these jointly-watched memory locations. Shared writable mappings are a common means of achieving high-performance inter-process communications.
- **Copy on write:** Copy on write is a lazy allocation technique. If a copy of a resource already in memory is requested, the request is satisfied by returning a mapping to the original resource. If one of the processes "sharing" the resource tries to write to it, the resource must be truly replicated in memory to allow the changes to be made to the new copy. So the memory allocation only takes place on the first write command.

For swappiness, we need only concern ourselves with the first two in the list: file pages and anonymous pages.

## Swappiness

Here's the description of swappiness [from the Linux documentation on GitHub](#):

"This control is used to define how aggressive (sic) the kernel will swap memory pages. Higher values will increase aggressiveness, lower values decrease the amount of swap. A value of 0 instructs the kernel not to initiate swap until the amount of free and file-backed pages is less than the high water mark in a zone.

The default value is 60."

That sounds like swappiness turns swap up or down in intensity. Interestingly, it states that setting swappiness to zero doesn't turn off swap. It instructs the kernel not to swap until certain conditions are met. But swapping can still occur.

Let's dig deeper. Here's the definition and default value of `vm_swappiness` [in the kernel source code file vmscan.c](#):

```
/*
 * From 0 .. 100. Higher means more swappy.
 */
int vm_swappiness = 60;
```

The swappiness value can range from 0 to 100. Again, the comment certainly sounds like the swappiness value has a bearing on how much swapping takes place, with a higher figure leading to more swapping.

Further on in the source code file, we can see that a new variable called `swappiness` is assigned a value that is returned by the function `mem_cgroup_swappiness()`. Some more tracing through the source code will show that the value returned by this function is `vm_swappiness`. So now, the variable `swappiness` is set to equal whatever value `vm_swappiness` was set to.

```
int swappiness = mem_cgroup_swappiness(memcg);
```

And [a little further down in the same source code file](#), we see this:

```
/*
 * With swappiness at 100, anonymous and file have the same priority.
 * This scanning priority is essentially the inverse of IO cost.
 */
anon_prio = swappiness;

file_prio = 200 - anon_prio;
```

That's interesting. Two distinct values are derived from `swappiness`. The `anon_prio` and `file_prio` variables hold these values. As one increases, the other decreases, and vice versa.

2%

The Linux swappiness value actually sets the ratio between two values.

## The Golden Ratio

File pages hold data that can be easily retrieved if that memory is freed. Linux can just read the file again. As we've seen, if the file data has been changed in RAM, those changes must be written to the file before the file page can be freed. But, either way, the file page in RAM can be repopulated by reading data from the file. So why bother adding these pages to the swap partition or swap file? If you need that data again, you might as well read it back from the original file instead of a redundant copy in the swap space. So file pages are not stored in swap. They're "stored" back in the original file.

With anonymous pages, there is no underlying file associated with the values in memory. The values in those pages have been dynamically arrived at. You can't simply read them back in from a file. The only way anonymous page memory values can be recovered is to store the data somewhere before freeing the memory. And that's what swap holds. Anonymous pages that you are going to need to reference again.

But note that for both file pages and for anonymous pages, freeing up the memory may require a hard drive write. If the file page data or the anonymous page data has changed since it was last written to the file or to swap, a file system write is required. To retrieve the data will require a file system read. Both types of page reclaim are costly. Trying to reduce hard drive input and output by minimizing the swapping of anonymous pages only increases the amount of hard drive input and output that is required to deal with file pages being written to, and read from, files.

As you can see from the last code snippet, there are two variables. One called `file_prio` for "file priority", and one called `anon_prio` for "anonymous priority".

- The `anon_prio` variable is set to the Linux swappiness value.
- The `file_prio` value is set to 200 minus the `anon_prio` value.

These variables hold values that work in tandem. If they are both set to 100, they are equal. For any other values, `anon_prio` will decrease from 100 towards 0, and `file_prio` will increase from 100 towards 200. The two values feed into a complicated algorithm that determines whether the Linux kernel runs with a preference for reclaiming (freeing up) file pages or anonymous pages.

You can think of `file_prio` as the system's willingness to free up file pages and `anon_prio` as the system's willingness to free anonymous pages. What these values don't do is set any kind of trigger or threshold for when swap is going to be used. That's decided elsewhere.

But, when memory needs to be freed, these two variables---and the ratio between them---are taken into consideration by the reclamation and swap algorithms to determine which page types are preferentially considered for freeing up. And that dictates whether the associated hard drive activity will be processing files for file pages or swap space for anonymous pages.

## When Does Swap Actually Cut In?

We've established that the Linux swappiness value sets a preference for the type of memory pages that will be scanned for potential reclamation. That's fine, but something must decide when swap is going to cut in.

Each memory zone has a high water mark and a low water mark. These are system derived values. They are percentages of the RAM in each zone. It is these values that are used as the swap trigger thresholds.

To check what your high and low water marks are, look inside the `/proc/zoneinfo` file with this command:

```
less /proc/zoneinfo
```

```
dave@howtogeek:~$ less /proc/zoneinfo
```

Each of the zones will have a set of memory values measured in pages. Here are the values for the DMA32 zone on the test machine. The low water mark is 13966 pages, and the high water mark is 16759 pages:

```
Node 0, zone DMA32
  pages free    16547
        min     11173
        low     13966
        high    16759
        spanned 520176
        present 520176
        managed 505667
        protection: (0, 0, 0, 0, 0)
  nr_free_pages 16547
  nr_zone_inactive_anon 5088
  nr_zone_active_anon 228881
  nr_zone_inactive_file 126856
  nr_zone_active_file 69332
  nr_zone_unevictable 4
  nr_zone_write_pending 82
  nr_mlock 4
  nr_page_table_pages 11138
  nr_kernel_stack 8920
  nr_bounce 0
```

- In normal running conditions, when free memory in a zone drops below the zone's low water mark, the swap algorithm starts scanning memory pages looking for memory that it can reclaim, taking into account the relative values of `anon_prio` and `file_prio`.
- If the Linux swappiness value is set to zero, swap occurs when the combined value of file pages and free pages are less than the high water mark.

So you can see that you cannot use the Linux swappiness value to influence swap's behavior with respect to RAM usage. It just doesn't work like that.

## What Should Swappiness Be Set To?

This depends on hardware, workload, hard drive type, and whether your computer is a desktop or a server. Obviously, this isn't going to be a one size fits all type of setting.

And you have to bear in mind that swap isn't just used as a mechanism to free up RAM when you're running out of memory space. Swap is an important part of a well functioning system, and without it, sane memory management becomes very difficult for Linux to achieve.

Changing the Linux swappiness value has an instant effect; you don't need to reboot. So you can make small adjustments and monitor the effects. Ideally, you'd do this over a period of days, with different types of activity on your computer, to try to find the closest to an ideal setting that you can.

These are some points to consider:

- Trying to "disable swap" by setting the Linux swappiness value to zero simply shifts the swap-associated hard drive activity to file-associated hard drive activity.



- If you have aging, mechanical hard drives, you might try reducing the Linux swappiness value to bias away from anonymous page reclamation and reduce swap partition churn. Of course, as you turn down one setting, the other setting increases. Reducing swap churn is likely to increase the file system churn. But your computer might be happier favoring one method over the other. Really, the only way to know for sure is to try and see.
- For single-purpose servers, such as database servers, you may get guidance from the suppliers of the database software. Very often, these applications have their own purpose-designed file cache and memory management routines that you'd be better to rely on. The software providers may suggest a Linux swappiness value according to machine specification and workload.
- For the average desktop user with reasonably recent hardware? Leave it as it is.

## How to Set the Linux Swappiness Value

Before you change your swappiness value, you need to know what its current value is. If you want to reduce it a little bit, the question is a little bit less than what? You can find out with this command:

```
cat /proc/sys/vm/swappiness
```

```
dave@howtogeek:~$ cat /proc/sys/vm/swappiness
60
dave@howtogeek:~$
```

To configure the swappiness value, use the [sysctl](#) command:

```
sudo sysctl vm.swappiness=45
```

```
dave@howtogeek:~$ sudo sysctl vm.swappiness=45
vm.swappiness = 45
dave@howtogeek:~$
```

The new value is used straight away, no reboot is required.

In fact, if you do reboot, the swappiness value will return to its default value of 60. When you have finished experimenting and have decided on the new value you wish to use, you can make it persistent across reboots by adding it to the `/etc/sysctl.conf` file. You can use whichever editor you prefer. Use the following command to edit the file with the nano editor:

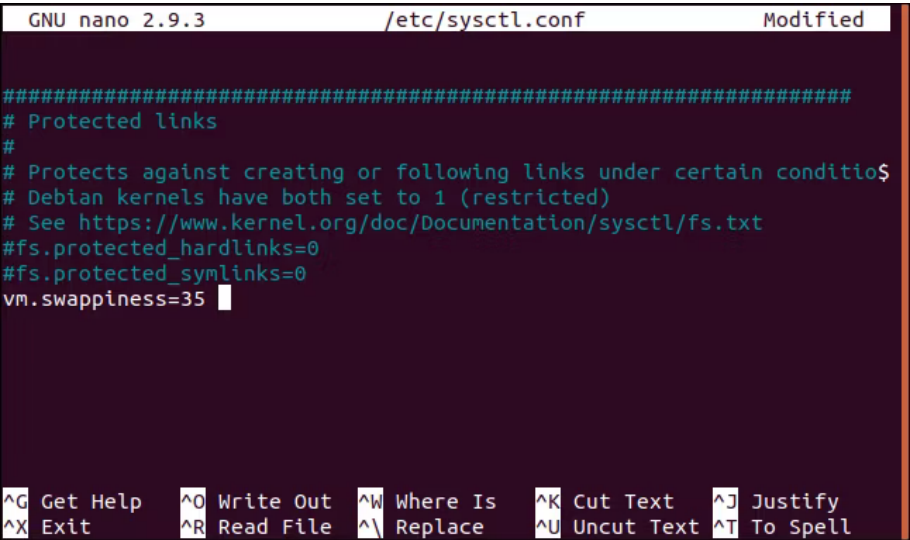
```
sudo nano /etc/sysctl.conf
```

```
dave@howtogeek:~$ sudo nano /etc/sysctl.conf
```



When nano opens, scroll to the bottom of the file and add this line. We're using 35 as the permanent swappiness value. You should substitute the value you wish to use.

```
vm.swappiness=35
```



To save your changes and exit from nano, press "Ctrl+O", press "Enter", and press "Ctrl+Z."

## Memory Management is Complex

Memory management is complicated. And that's why, for the average user, it is usually better to leave it up to the kernel.

It's easy to think you're using more RAM than you are. Utilities like `top` and `free` can give the wrong impression. Linux will use free RAM for a variety of its own purposes, such as disk caching. This artificially elevates the "used" memory figure and reduces the "free" memory figure. In actual fact, the RAM used as disk cache is flagged as both "used" and "available" because it can be reclaimed at any time, very quickly.

To the uninitiated that might look like swap isn't working, or that the swappiness value needs changing.

As always, the devil is in the detail. Or, in this case, the daemon. The kernel swap daemon.

	Linux Commands
Files	<a href="#">tar</a> · <a href="#">pv</a> · <a href="#">cat</a> · <a href="#">tac</a> · <a href="#">chmod</a> · <a href="#">grep</a> · <a href="#">diff</a> · <a href="#">sed</a> · <a href="#">ar</a> · <a href="#">man</a> · <a href="#">pushd</a> · <a href="#">popd</a> · <a href="#">fsck</a> · <a href="#">testdisk</a> · <a href="#">seq</a> · <a href="#">fd</a> · <a href="#">pandoc</a> · <a href="#">cd</a> · <a href="#">\$PATH</a> · <a href="#">awk</a> · <a href="#">join</a> · <a href="#">jq</a> · <a href="#">fold</a> · <a href="#">uniq</a> · <a href="#">journalctl</a> · <a href="#">tail</a> · <a href="#">stat</a> · <a href="#">ls</a> · <a href="#">fstab</a> · <a href="#">echo</a> · <a href="#">less</a> · <a href="#">chgrp</a> · <a href="#">chown</a> · <a href="#">rev</a> · <a href="#">look</a> · <a href="#">strings</a> · <a href="#">type</a> · <a href="#">rename</a> · <a href="#">zip</a> · <a href="#">unzip</a> · <a href="#">mount</a> · <a href="#">umount</a> · <a href="#">install</a> · <a href="#">fdisk</a> · <a href="#">mkfs</a> · <a href="#">rm</a> · <a href="#">rmdir</a> · <a href="#">rsync</a> · <a href="#">df</a> · <a href="#">pgp</a> · <a href="#">vi</a> · <a href="#">nano</a> · <a href="#">mkdir</a> · <a href="#">du</a> · <a href="#">ln</a> · <a href="#">patch</a> · <a href="#">convert</a> · <a href="#">rclone</a> · <a href="#">shred</a> · <a href="#">srm</a> · <a href="#">scp</a> · <a href="#">gzip</a> · <a href="#">chattr</a> · <a href="#">cut</a> · <a href="#">find</a> · <a href="#">umask</a> · <a href="#">wc</a> · <a href="#">tr</a>
Processes	<a href="#">alias</a> · <a href="#">screen</a> · <a href="#">top</a> · <a href="#">nice</a> · <a href="#">renice</a> · <a href="#">progress</a> · <a href="#">strace</a> · <a href="#">systemd</a> · <a href="#">tmux</a> · <a href="#">chsh</a> · <a href="#">history</a> · <a href="#">at</a> · <a href="#">batch</a> · <a href="#">free</a> · <a href="#">which</a> · <a href="#">dmesg</a> · <a href="#">chfn</a> · <a href="#">usermod</a> · <a href="#">ps</a> · <a href="#">chroot</a> · <a href="#">xargs</a> · <a href="#">tty</a> · <a href="#">pinky</a> · <a href="#">lsof</a> · <a href="#">vmstat</a> · <a href="#">timeout</a> · <a href="#">wall</a> · <a href="#">yes</a> · <a href="#">kill</a> · <a href="#">sleep</a> · <a href="#">sudo</a> · <a href="#">su</a>
The Best Tech Newsletter Around	
<a href="#">hw</a> · <a href="#">shutdown</a> · <a href="#">reboot</a> · <a href="#">halt</a> · <a href="#">poweroff</a> · <a href="#">passwd</a> · <a href="#">lscpu</a> · <a href="#">crontab</a> · <a href="#">date</a> · <a href="#">bg</a> · <a href="#">fg</a> · <a href="#">pidof</a> · <a href="#">nohup</a> · <a href="#">pmap</a>	

Email Address

[cat](#) · [ping](#) · [traceroute](#) · [ip](#) · [ss](#) · [whois](#) · [fail2ban](#) · [omni](#)

Networking

[nmap](#) · [ftp](#) · [curl](#) · [wget](#) · [who](#) · [whoami](#) · [w](#) · [iptables](#) · [ssh](#) · [keygen](#) · [ssh](#)

By subscribing, you agree to our [Privacy Policy](#) and may receive occasional deal communications; you can unsubscribe anytime.

SUBSCRIBE

RELATED: [Best Linux Laptops for Developers and Enthusiasts](#)

Share

Tweet

Share

Share

Share

Copy

Email

Related Topics

- LINUX
- FEATURES
- LINUX & MACOS TERMINAL
- LINUX

About The Author

Dave McKay (400 Articles Published) [in](#)

Dave McKay first used computers when punched paper tape was in vogue, and he has been programming ever since. After over 30 years in the IT industry, he is now a full-time technology...

