

How to Create a Swap File on Linux

Dave McKay

12-15 minutes



zentilia/Shutterstock.com

Add swap space to a Linux computer, or increase the swap space that's already present, without messing about with partitions. We show you the easy way to tailor your swap space.

How to Speed Up a Slow PC

0 seconds of 1 minute, 13 secondsVolume 0%

Swap Files vs. Swap Partitions

There are several scenarios where you might want to increase existing or add new swap space to your Linux computer.

- Perhaps your swap space is frequently running at maximum or close to maximum.
- It's easy to click the wrong option during the installation process and to decline adding swap to your system inadvertently.

- Maybe you previously decided that you had so much [random access memory](#) (RAM) you didn't need any swap, and you've changed your mind.
- Sometimes you inherit the administration of a system that has no swap, for reasons you'll never be able to discover.

The simple solution to all of these is to add a [swap file to your computer](#). This is a special file, pre-allocated and reserved for use as swap space. A swap file will work in tandem with any existing swap you may have, whether that is a swap file or a swap partition.

At one time, there was a performance hit for using a swap file compared to a swap partition. That's no longer the case with improvements in the performance of mechanical (spinning) hard drives and more efficiency in the swap functions within the Linux operating system. In fact, some Linux distributions now default to creating swap files rather than swap partitions.

Swap isn't just used as a means to free up RAM when you're running low on memory. It's an important part of a well functioning system. Without any swap, sane memory management becomes very difficult for the kernel to achieve. Let's look at the easy way to add some swap space.

Before We Dive In: Btrfs and SSDs

There are two points we'd like to discuss quickly.

The [Btrfs file system](#) has certain caveats about swap files. At one time, there was a conflict between the [copy-on-write](#) nature of Btrfs, which wanted to operate in one way and swap files that needed to operate in another. Some functions that swap files depend on were not implemented, and some assumptions that were made about block numbering within the swap files did not hold true with Btrfs. So swap files were not supported.

Since kernel 5.0, you [can have swap files in Btrfs file systems](#) if they are set up with the following requirements:

- No copy-on-write (NOCOW) files.
- They're not compressed.
- They don't straddle different hard drives.

Most readers will be using the [default ext4 file system](#), so this won't be a concern to them.

RELATED: [*Which Linux File System Should You Use?*](#)

When [Solid-State Drives \(SSDs\)](#) were first available, there was a concern about using them in situations that had frequent file system writes. People were warned off from putting swap space on SSDs, and even to avoid system logging to SSDs.

The Best Tech Newsletter Anywhere

Join **425,000** subscribers and get a daily digest of features, articles, news, and trivia.

By submitting your email, you agree to the [Terms of Use](#) and [Privacy Policy](#).

This is much less of a concern nowadays, and [many SSDs that are on sale have life expectancies that will outlast most PCs](#). A swap file on an SSD will have a far better performance than a swap partition on a mechanical hard drive.

RELATED: [*How Long Do Solid State Drives Really Last?*](#)

Checking Existing Swap Space

Look before you leap. Let's check what swap space is available on your computer. You can do this two ways, and we'll use both. [The free command will display the used and free memory](#). The -h (human readable) option will cause free to use sensible units when it displays the memory values.

free -h

```
dave@Ubuntu:~$ free -h
              total        used        free      shared  buff/cache
available
Mem:          1.9G          935M          237M           19M           817M
```

```
881M
Swap: 0B 0B 0B
dave@Ubuntu:~$
```

The output from `free` shows that there is no swap space configured on this machine.

Swap is never discussed without RAM and free RAM cropping up. So it's worth noting that the free RAM is given as 237 MB. Don't mistake that for the total of RAM available for use. That value is provided by the "available" figure, which is 881 MB.

Linux uses free RAM for its own purposes, such as file caching and kernel buffers. The amount of RAM dedicated to that is the "buff/cache" figure, which is 871 MB. But that memory is still regarded as—and counted as—"available." The contents of the "buf/cache" RAM can be discarded immediately and used by any application that needs some memory.

Another way to check if swap space is available is to use the `swapon` command. The `--show` option doesn't make any [changes to the swap on your computer](#). It only provides statistics.

```
swapon --show
```

```
dave@Ubuntu:~$ swapon --show
```

If there is no output from this command, there's no swap configured.

If these commands had revealed some swap space is already configured, the size of the existing swap space should be factored into decisions regarding the size of the swap file you're going to create.

How Much Swap Space Do I Need?

The traditional response was "twice the amount of RAM you have." But this was coined when computers used to have very limited RAM. As RAM has become cheaper, and programs and games

more demanding of memory, PC specifications have adjusted accordingly. Home PCs with 32 GB of RAM are not uncommon nowadays. And you're not going to allocate 64 GB of hard drive space to swap space if you've got 32 GB of RAM. That's plainly excessive.

The amount of swap you need is as an incendiary subject, comparable to "which is the best editor." One of the most sensible discussions we've seen on this topic is in the [Ubuntu swap FAQ](#). It's a short and commonsense approach (although, like many people, they misunderstand [how swappiness works on Linux](#)). There's a handy table that shows a recommended amount of swap space for the amount of RAM your system has, and whether you hibernate your computer or not.

And the good news is, it doesn't really matter what value you pick. We can always remove a swap file and replace it with a bigger one or, for that matter, a smaller one. Or you could just add another swap file.

Pick [a swap file size from the table](#), and run it for a while. Monitor your system's use of the swap space. If fine-tuning is required, changes are easily made. With swap files, It's a two-minute job. Compare that to adjusting partitions on a live Linux computer.

RELATED: [***What Is Swappiness on Linux? \(and How to Change It\)***](#)

Creating the Swap File

You shouldn't use the `fallocate` command [to create your swapfile](#). This is from the man page for `swapon`:

The swap file implementation in the kernel expects to be able to write to the file directly, without the assistance of the file system.

This is a problem on files with holes or on copy-on-write files on file

systems like Btrfs. Commands like `cp(1)` or `truncate(1)` create files with holes. These files will be rejected by `swapon`.

Preallocated files created by `fallocate(1)` may be interpreted as files with holes too depending of the filesystem. Preallocated swap files are supported on XFS since Linux 4.18.

The most portable solution to create a swap file is to use `dd(1)` and `/dev/zero`.

So, although `fallocate` is faster, we'll use `dd` to [create the swap file](#). The machine used to research this article has two GB of RAM. We're going to create a one GB swap file.

The options are:

- **if**: The input file. In this example, we're using `/dev/zero` which will provide a stream of zero bytes.
- **of**: The output file. We're going to create a file in the root directory, called `swapfile`.
- **bs**: The block size in bytes. This specifies how many bytes to read from the input file and to write to the output file, at a time.
- **count**: How many blocks to read and write. Multiply this number by the `bs` value to get the file size.

```
sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
```

```
dave@Ubuntu:~$ sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
```

Some statistics are provided when the file is created.

```
dave@Ubuntu:~$ sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
1048576+0 records in
1048576+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.66996 s, 643 MB/s
dave@Ubuntu:~$
```

We can see the number of blocks (records) that were written to the file, the size of the file, the time taken to create the file, and the effective data transfer rate.

Use the `ls` command to see the file in the root directory:

`ls /`

```
dave@Ubuntu:~$ ls /
bin      etc      lib      mnt      run      swapfile  var
boot     home     lib64     opt      sbin     sys       vmlinuz
cdrom    initrd.img  lost+found  proc    snap     tmp       vmlinuz.old
dev      initrd.img.old  media      root    srv      usr
```

Preparing the Swap File

We need to [prepare the swap file](#) with the `mkswap` command before it can be used. We don't need to provide any parameters to `mkswap` other than the path and name of the file:

```
sudo mkswap /swapfile
```

```
dave@Ubuntu:~$ sudo mkswap /swapfile
mkswap: /swapfile: insecure permissions 0644, 0600 suggested.
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=36874029-1079-426b-a93a-cb1997f1bf0a
dave@Ubuntu:~$
```

The file is prepared for use as a swap file. Note the warning about file permissions. We'll need to change those so that the root user is the only one who can read and write to the swap file.

Using the Swap File

The default permissions are too liberal, we need to restrict them so that only root can use the swapfile. [Use chmod to change the file permissions](#):

```
sudo chmod 600 /swapfile
```

```
dave@Ubuntu:~$ sudo chmod 600 /swapfile
```

This removes all permissions from the file group members and

others, but allows the file owner, root, to read and write to the file.

RELATED: [How to Use the chmod Command on Linux](#)

We need to use the swapon command to let Linux know there is a new swap file available to use. We only need to provide the path and the filename:

```
sudo swapon /swapfile
```

```
dave@Ubuntu:~$ sudo swapon /swapfile
dave@Ubuntu:~$
```

The swap file is now active.

Adding the Swap File to fstab

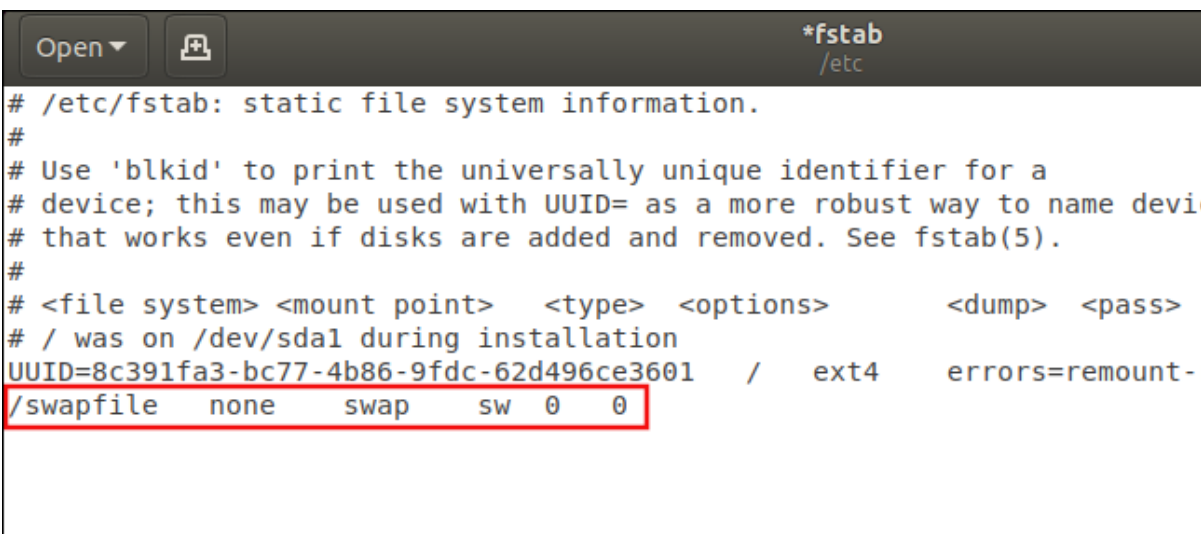
To make sure your swap file is available after a reboot, [add it to the /etc/fstab file](#). You can use any text editor you prefer, but we'll show the process [using the graphical Gedit text editor](#).

```
sudo gedit /etc/fstab
```

```
dave@Ubuntu:~$ sudo gedit /etc/fstab
```

The line we need to add to the bottom of the file is:

```
/swapfile none swap sw 0 0
```



```
*fstab
/etc

# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devi
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=8c391fa3-bc77-4b86-9fdc-62d496ce3601 / ext4 errors=remount-
/swapfile none swap sw 0 0
```

The fields are:

- **File system:** The path and name of the swap file.
- **Mount point:** The file isn't mounted like a file system, so the entry

is “none.”

- **Type:** This is “swap.”
- **Options:** At boot time `swapon -a` (start all devices marked as swap) will be called from one of the boot scripts. This option tells Linux to treat this entry as a swap resource that should come under the control of that `swapon -a` command. It is common to see “defaults” used here because there is a mistaken belief amongst some Linux users that this field is ignored. As we shall see, that is not the case. So it makes sense to use the correct option.
- **Dump:** This can be set to zero. It is irrelevant in this case.
- **Pass:** This can be set to zero. It is irrelevant in this case.

Save the changes and close the editor.

RELATED: [How to Write an fstab File on Linux](#)

Checking Swap Usage

To see if your swap space is being used, use the `swapon` command with the `--show` option:

`swapon --show`

```
dave@Ubuntu:~$ swapon --show
NAME      TYPE  SIZE USED PRIO
/swapfile file 1024M 524K  -2
dave@Ubuntu:~$
```

The columns are:

- **Name:** The name of the swap partition or swap file.
- **Type:** The type of swap device.
- **Size:** The size of the swap resource.
- **Used:** The amount of used swap space.
- **Prio:** The priority of this swap space.

The Swap Space Priority

Each swap space is allocated a priority. If you don't provide one, one is automatically allocated. Automatically allocated priorities are always negative. The range of priorities that can be manually allocated is 0 to 32767. Swap resources with higher priorities are used first.

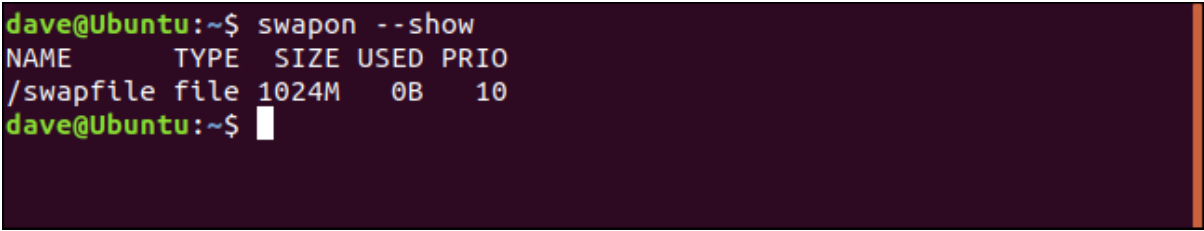
If more than one swap space has the same priority they are used alternately until they are both full, then the system looks for the swap space with the next lowest priority. If you only have a single swap space then the priority is irrelevant of course. But we'll change the priority of the swapfile we've created to demonstrate how to do it.

To set a priority, add the `pri= (priority)` option to the `/etc/fstab` entry. Edit the line you added to `/etc/fstab` to look like this:

```
/swapfile none swap sw,pri=10 0 0
```

That is, add `pri=10` to the options field, separated from the "sw" with a comma. Do not leave any spaces between the "sw", the comma, and the "pri=10." Reboot your computer and use the `swapon --show` command:

```
swapon --show
```



```
dave@Ubuntu:~$ swapon --show
NAME      TYPE  SIZE USED PRIO
/swapfile file 1024M  0B   10
dave@Ubuntu:~$
```

The priority of this swap space has been elevated to 10. Which is proof positive that the options field in the `/etc/fstab` entry is not ignored.

Swap Space Made Easy

Cutting through the exposition and explanation, we can create a new swap file as easily and quickly as this:

```
sudo dd if=/dev/zero of=/swapfile2 bs=1024 count=104857
```

```
sudo mkswap /swapfile2
```

```
sudo chmod 600 /swapfile2
```

```
sudo swapon /swapfile2
```

And let's check that it worked:

```
swapon --show
```

```
dave@Ubuntu:~$ sudo dd if=/dev/zero of=/swapfile2 bs=1024 count=104857
6
1048576+0 records in
1048576+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.73797 s, 618 MB/s
dave@Ubuntu:~$ sudo mkswap /swapfile2
mkswap: /swapfile2: insecure permissions 0644, 0600 suggested.
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=066f33e3-81c7-4c7a-9ac5-49b148518df6
dave@Ubuntu:~$ sudo chmod 600 /swapfile2
dave@Ubuntu:~$ sudo swapon /swapfile2
dave@Ubuntu:~$ swapon --show
NAME      TYPE      SIZE USED PRIO
/swapfile file 1024M 268K  25
/swapfile2 file 1024M   0B  -2
dave@Ubuntu:~$
```

If you want to make that permanent drop, it into your `/etc/fstab` file.

Boom. Job done.

READ NEXT

- › [How to Use SUID, SGID, and Sticky Bits on Linux](#)
- › [How to Turn Off RTT on iPhone](#)
- › [Try These Last Minute Halloween Smart Home and Tech Tricks](#)
- › [How to Restart a PS4](#)
- › [How \(and Why\) to Use Your Mac's Background Sounds Feature](#)
- › [You Can Use Power Tool Batteries on Your Dyson Stick Vacuum](#)
- › [How to Test a Suspicious Link Before Clicking It](#)