

Increase your available swap space with a swap file

Problem:

My server runs out of RAM and also Swap. How can I increase this Swap volume ?

You've finally configured every aspect of your Linux system. All of your devices function, and everything is configured just the way you like it. At least you think so, until you start running out of memory when you have OpenOffice.org and lots of browser tabs open simultaneously. You realize you should have specified a larger swap partition during your install. Does this mean you have to install again from scratch? Happily, no. Here's how to set up some extra swap space on one of your existing partitions.

There's always more than one way to maintain your Linux system. Instead of creating a swap file, you could instead resize and or reshuffle your partitions with parted or its graphical front end QtParted. However, resizing your partitions is a much more severe change than simply adding a file to one of them.

To start off, see how much swap space you already have. At a command line, type `swapon -s` (you might need to prepend `/sbin/` if you're not root). The command should produce a message that looks something like this: `Filename Type Size Used Priority /dev/hda2 partition 128044 92472 -1`

The numbers under "Size" and "Used" are in kilobytes. Uh oh. On this system we've used about 92MB of our measly 128MB swap partition -- and we're not even running OpenOffice.org yet. We need an auxiliary swap file.

Let's figure out where to put it. Running `df -m` (short for "disk free") from a command line should produce output something like this: `Filesystem 1M-blocks Used Available Use% Mounted on /dev/hda1 11443 6191 5252 55% /`

The `-m` switch we used provided us with output in megabytes. Under the "Available" column we have approximately 5GB of free space on our root partition. Let's steal 512MB of that for our auxiliary swap file. You might want more or less, depending on your memory needs, how much swap space you already have available, and how much free disk space you have. The general rule of thumb for swap size is that your total available swap space should be around double your RAM size. If you have additional partitions, and one of those is a better candidate than the `/` partition, feel free to use it instead.

You may have heard before that "everything's a file in Unix and Linux." Device files (like our `/dev/hda1` partition) are a special type of file. What we're about to do is create a plain-jane, non-special file, prepare it for swapification, and then tell Linux to use it when it runs out of space on its regular swap device (file).

A brief word of warning. Back up your important data before proceeding. If you carefully follow the steps below you should be fine, but it's always better to be safe.

In order to create our supplementary swap file, we're going to use the `dd` (data dump) command. You'll need to become root to perform the next few steps. In a regular terminal type `su -` and enter your root password. When you're ready, carefully type: `dd if=/dev/zero of=/extraswap bs=1M count=512`

replacing 512 with the number of megabytes you want in your auxiliary swap file. `if=` and `of=` are short for infile and outfile. The `/dev/zero` device file will give us zeroes to be written to the output file. If you want this file on a different partition, say your `/var` partition, you would replace `/extraswap` with `/var/extraswap`.

Now we have a file the size we want on disk, and we can prepare it for use as a swap partition. We'll use the `mkswap` command to make our file swap-consumable for the Linux kernel. Again as root, carefully type: `mkswap /extraswap`

To turn on our swap file, we run `swapon /extraswap`. Now when we run `swapon -s` we should see our existing swap partition and our new swapfile. Also, the `free` command should show an increase in total swap space.

But we're still not done yet. If we reboot our machine now, our new swapfile won't be active, and we'll have to run `swapon /extraswap` again. In order to make things more permanent, we'll need to edit our `/etc/fstab` file.

First, make a copy of the file. (You'll see why shortly.) Something like this should do the trick: `cp /etc/fstab /etc/fstab.mybackup`

Now open `/etc/fstab` in your favorite text editor and find a line about your swapfile that looks something like this: `/dev/hda2 none swap sw 0 0`

You'll need another line like that underneath it pointing to your new swap file. Replace the first column with the location of

your new swap file. For our example, the new line should look like this:`/extraswap none swap sw 0 0`

Save the file. Mistaken changes to `/etc/fstab` could render your system unbootable, so just to make sure you didn't accidentally change anything else in `/etc/fstab`, run `diff /etc/fstab.mybackup /etc/fstab` to check for differences. That should output only the single line you added, with a `>` sign in front of it. If you see anything else in diff's output, edit `/etc/fstab` again, fix it, and run the above diff command again.

Congratulations! You should now have more swap space than you did before.

Was this an ideal solution? Hardly. Linux nuts would call this a "dirty hack." When swapping with your new auxiliary swap file your system will be working a little harder than when it's swapping to a dedicated swap partition. In order to access the swapfile, the Linux kernel will have to first talk to the filesystem that the swapfile is on before it can get the information it wants from the swapfile itself. When swapping directly to a partition, the kernel doesn't have to take this extra step.

In practical terms, there's a minimal performance hit from this extra step. By the time you've run out of RAM and are beginning to swap, you're already suffering a massive performance hit. After your original swap partition is full and you're spilling into your auxiliary swap file, your system should be suffering badly enough that the added performance hit will be completely imperceptible.

In order to avoid this sort of problem entirely with your next install, using Linux's Logical Volume Manger is probably a good idea, and there are other Linux memory management techniques. Of course the ideal solution is to just install additional RAM.