Linux compress pdf file size

Click to viewDo you've already been sent a simple ZIP archive, you need to create and share your own compressed files, or you've stared down a barrel of some obscure archive format that you've never seen before, having the right file compression application in your corner is a must. Earlier this week, we asked you to share your favorite file compression tool, and more than 500 comments later we're back with the top five answers. Read on for a closer look at the five best file compression tools, then get ready to punch chad for the app you like best. 7-Zip (Windows/Linux, Free) 7-Zip is a free, open-source file archive tool with a replacement interface but a powerful set of features. With support for most popular compression formats (and quite a few not-so-popular), this lightweight, open source option does the job quickly and seamlessly. While some 7-Zip users complain about their replacement interface, others are happy with the no-nonsense 7-Zip approach and fast operation. IZArc (Windows, Freeware)G/O Media can get commissionIZArc is a compression tool that can take home the price for the most supported reading and writing formats for this Hive Five. IZArc is also the only featured archiver unlike PeaZip that distributes the portable version on its website (although third parties have made other applications portable-like 7-Zip Portable). Users go to IZArc for its attractive interface and its low price tag. IZArc is freeware, but donations are accepted. WinRAR (Windows, Shareware)WinRAR is a powerful file compression and decompression tool that has existed since 1993. As the first result in Google search for RAR, this is probably the first option most of us came across when we came across our first RAR file. This means that WinRAR supports a wide range of formats. It is also one of the few archivers able to write RAR archives – although overall it is limited to creating only RARs or ZIPs. WinRAR costs quite a steep $29 per license, but several users are happy to suffer through annoying screens to avoid costs. PeaZip (Windows and Linux, Free)PeaZip is a free and open-source archive manager that supports a bunch of formats. Unlike its open-source sister, 7-Zip, PeaZip also has a very attractive interface, from the main application interface to the desktop icons it uses when you set it as the default compression tool. Like IZArc, it's also available in a portable version – so even if you don't go with it for default settings, it's worth to throw away on a flash drive just in case you need a little compression on the road. Unarchiver (Mac OS X, Freeware)Unarchiver is the built-in default file compression tool for Mac OS X. Unlike Windows, which only supports zip out-of-the-box format, Unarchiver handles most major formats. Catch: Unarchiver is a read-only app, so if you're on a Mac and want to write to more obscure archives other than ZIP, you may need to add another tool to your arsenal. Most OS X users, however, are happy to stick to hold Unarchiver for all their decompression needs. Now that you've seen the best, it's time to vote. This week's honorable mentions go out to jZip and ALZip. Whether your chosen app has created a shortlist, let's hear about it in the comments. Mozilla has released an updated version of its JPEG compression tool, which shaves down file sizes by 5 percent, a small number, but one that is significant for image-intensive web services like Facebook.The tool, called mozjpeg 2.0, will eventually reduce page load time and ultimately create a better user experience for image hosting sites, wrote Josh Aas, senior technology strategist for Mozilla on his blog. Facebook began testing the tool and donated $60,000 for its further development, Aas wrote. JPEG has been in use for more than 20 years, and most images on the Internet are served in this format. It is a lossy or compressed image format that aims to delete some data to reduce file size but preserve the integrity of the photo as much as possible. On average, both basic and progressive jpeg files are reduced by 5 percent mozjpeg, Aas wrote. The previous iteration of mozjpeg only improved compression for progressive JPEG, Aas wrote. Mozjpeg is based on the libjpeg-turbo library, which is used to decode JPEG in Firefox. But mozjpeg consume more computing power than libjpeg-turbo when compressed, Aas wrote. As a result, we recommend using libjpeg-turbo for the standard JPEG library and all decoding tasks, he wrote. Use mozjpeg when creating JPEG for the web. Note: When you buy something by clicking on the links in our articles, we can get a small commission. Read our affiliate link policy for more information. Fatmawati Achmad Zaenuri/Shutterstock When you use the Linux du command, you get both the actual disk usage and the actual file size or directory. We'll explain why these values aren't the same. Actual disk usage and actual size File size and the space it takes up on the hard disk are rarely the same. Disk space is allocated in blocks. If the file is smaller than a block, it is still allocated an entire block because the file system does not have a smaller real estate drive. If the file size is not an exact multiple of blocks, the space it uses on the hard disk must always be rounded up to the next entire block. For example, if a file is larger than two blocks but less than three, it still takes three blocks of space to store it. Two measurements are used in relation to file size. The first is the actual file size, which is the number of bytes of content that make up the file. The second is the efficient file size on your hard drive. This is the number of file system blocks that are necessary to save this file. Example Let's look at a simple example. We redirect one character to a file to create a small file: echo 1 > geek.txt Now we use a long format listing, ls, look at the length of the file: ls -l geek.txt length is the numeric value that follows the dave dave item, which is two bytes. Why is it two bytes when we sent only one character to the file? Let's see what's going on inside the file. We use the hexdump command, which gives us the exact number of bytes and allows us to see non-print characters as hexadecimal values. We will also use the -C (canonical) option to force the output to display hexadecimal values in the output body, as well as their alphanumeric character equivalents: hexdump -C geek.txt Output shows us that starting with the offset of 00000000 in the file is a byte that contains a hexadecimal value of 31, and one that contains a hexadecimal value of 0A. The right side of the output represents these values as alphanumeric characters whenever possible. A hexadecimal value of 31 is used to represent the digit one. A hexadecimal value of 0A is used to represent a line character that cannot be displayed as an alphanumeric character, so it appears as a period (.) instead. The line feed character is added by an echo . By default, echo starts a new line after the text it needs to write to the terminal window appears. This coincides with the output from ls and corresponds to the length of the two-byte file. RELATED: How to use ls Command for the list of files and directories on Linux Now, we will use the du command to look at the file size: du geek.txt Says the size is four, but four of what? There are blocks, and then there are blocks when du reports file sizes in blocks, the size it uses depends on several factors. You can specify what block size to use at the command line. If you do not force the use of a specific block size, it follows a set of rules that decide which one to use. First, it checks the following environment variables: DU_BLOCK_SIZE BLOCK_SIZE BLOCKSIZE if any of them exist, the block size is set and the du check stops. If none are set, i default block size to 1,024 bytes. If the environment variable called the environment variable is not POSIXLY_CORRECT. If this is the case, i default block size to 512 bytes. So, how do we know which one is being used? You can check each environment variable to resolve it, but there is a faster way. Compare the results with the size of the block used by the file system. To determine the size of the block that the file system uses, we will use tune2fs. Then we use the option -l (list of superblocks), release the output with grapefruit, and then print the lines containing the word Block. In this example, we look at the file system on the first partition of the first hard drive, sda1, and we have to use sudo: sudo tune2fs -l /dev/sda1 | Grep Block The size of the file system block is 4,096 bytes. If we break it down according to the result we obtained from du (four), it shows that the size of the block du is 1,024 bytes. Now we know a few important things. First, we know that the smallest amount of real estate file system that can be dedicated to storing a file is 4096 bytes. This even our small double-byte file kills 4 KB of hard drive space. The second thing to keep in mind is the application dedicated to reporting on the hard drive and file system statistics, such as du, ls, and tune2fs, may have different ideas about what the block means. tune2fs reports the actual size of file system blocks, while ls and du can be configured or forced to use other block sizes. These block sizes are not intended to be related to the size of the file system block. they are just pieces that use these commands in their output. Finally, other than using different block sizes, responses from du and tune2fs convey the same meaning. The result of tune2fs was one block of 4,096 bytes, and the result du was four blocks of 1,024 bytes. Using du No parameters or command-line options du lists the total amount of disk space that the current directory uses and all subdirectresses. Let's look at the example: du Size is given in the default block size of 1,024 bytes per block. The whole tree of the subdirectory is run through. Use du on a different directory If you want to du a message about a directory other than the current directory, you can pass the path to the directory at the command line: du ~/.cach/evolution/ Using du on a Specific File If you want to report the du to a specific file, pass the flow path to the file flow at the command line. You can also pass a sample of the environment to the selected file group, such as *.txt: du ~/.bash_aliases Reporting on Files in Directories To have a du report on files in the current directory and subdirectories, use -a (all files): du -a For each directory, the size of each file is indicated, as well as the sum for each directory. Limiting the depth of a directory tree You can say du list the directory tree to a certain depth. To do so, use the -d (maximum depth) option and enter a depth value as a parameter. Note that all subdirectors are scanned and used to calculate reported totals, but not all are listed. To set the maximum directory depth at one level, use this command: du -d 1 Output shows the total size of this subdirectory in the current directory, and also provides a sum for each. To list directories one level deeper, use the following command: du -d 2 Set block size: You can use the block option to set the block size for du for the current operation. To use a single-byte block size, use the following command to obtain exact directory and file sizes: du --block=1 To use a one-megabyte block size, you can use the -m (megabyte) option, which is the same as --block=1M: du -m To report sizes in the most appropriate block size based on the disk space used by the directory and files, use the -h (human-readable) option: du -h To view the apparent file size rather than the amount of hard disk space used to store the file, use --apparent-size: du --apparent-size with -a (all) and look at the apparent size of each file : du --apparent size size Each file is listed, along with its apparent size. View totals only: Use the -s (summarize) option to report only the sum of the directory. You can also combine this with other options, such as -h (human-readable): du -h -s Here, we use it with --apparent-size: du --apparent-size -s Times Modification View To view the time and date of the last change, use --time: du --time -d 2 Strange results? If you see strange results from du , especially when cross-referenced output from other commands, this is usually due to different block sizes on which different commands can be set, or those on which they are default. This may also be due to differences between the actual file sizes and the disk space needed to store them. If you need to match the output of other commands, experiment with --block in du. Du.