I'm not robot

reCAPTCHA

Continue

Command to search for keywords

Not sure exactly you want to express yourself in emoji? Emoji is a simple small command line tool that allows you to crank out a sentence and easily search for emojis that fit the bill. Emoji runs from Node.js, so you'll need that installed first. Once it's taken care of, all you need to do is type emoji followed by the text you want to search for, it would be the emoji I'm hungry and Emoji does the rest. It's certainly a bit silly, but it's also an easy way to find the emoji you're looking for. Emoji | GitHub In this section: Application Reports Use Keyword Search to isolate results based on predefined search criteria fields (All, Exact, Any, or Exclusion). For example: All these Words: Search the term orange cross blood bank will retrieve the results that have all these words anywhere in the record in any order. Exactly Word (s): Search the term orange cross blood bank will retrieve the results that have the exact phrase: orange cross blood bank. Any of these Words (s): Search the term orange cross blood bank will retrieve the results that have any of the orange or cross or blood words or bank or any combination of those words. Exclude these words: The search term orange cross blood bank will take over results that DO NOT include the words orange or cross or blood or bank or any combination of these words. For a more meaningful set of results, the exclusion of these Word(s) fields should be used in combination with the other fields (All, Exact, or Any). To search for DATE, enter the date in the mm/dd/yyyy format in double quotation marks: 08/14/2017. Note: The Exact Word(s) field does not require double quotation marks for data: 14.08.2017. Back to Top Fatmawati Achmad Zaenuri/Shutterstock.com Linux offers six different ways to search, and each has its own merits. We'll demonstrate to use find, locate, who, where it is, what it is, and by the way. Each excels at different tasks; here's to choose the right tool for the job. You are spoiled for choice when it comes to commands for searching and finding in Linux. Why so many? Well, each has their specialties and works better than the others under certain circumstances. You could think of them as some kind of Swiss knife for the search. We'll look at each blade one at a time and find out its strengths. Find Command Behavior of the Find command is difficult to determine by trial and error. Once you understand the syntax, start appreciating its flexibility and strength. The easiest way to use find is to just find and hit enter. find used in this way find behaving like LS, but lists all the files in the directory and those in subdirectories. Some deployments to find require you to put. for the current directory. If this is the case with the Linux version, use the following command: find . To find the search from the root folder you would use this command: find / To start the search from you. Your. Folder use this command: find ~ Find ~ with file patterns to find it to be little more than a self-recursive version of ls, you must provide it something to search for. We can provide file names or file templates. Models make use of wildcard characters where * means any string and characters? means any unique character. Models must be quoted to function correctly. It's easy to forget to do this, but if you don't quote the wildcard model find you won't be able to properly carry out the command you gave it. With this command, we will search the current folder for files that match the *.*s model. This means any file name that has a file extension that ends in s. We use the -name option to say that we are either passing a file name or a file name model. Find. -name *.*s find returns these matching files. Note that two of the file extensions have two characters and one has three characters. This is because I used the *.*s model. If we only wanted the two character file extensions, would we have used *.? S. If we had known in advance that we were looking for JavaScript .js files we could have been more specific in our file model. Also note that you can use unique quote marks to frame the model, if you prefer. Find. -name * .js This time find only reports on JavaScript files. Ignore case by find If you know the name of the file you want to find, you can forward it to find it instead of a pattern. You don't need to enclose the file name in quotation marks if there are no wildcard characters in it, but it's a good practice to do it all the time. This means you won't forget to use them when you need them. Find. - The name Yelp.js who didn't return anything. But it's weird, we know the file has to be there. Let's try again and tell Find to ignore the case. We do this using the option -iname (ignore the name of the case) find. -iname Yelp.js That was the problem, the file name starts with a lower letter y, and I was looking with a y capital letters. Recourse subdirectories with find A great thing about finding is how you search recursively through subdirectories. Let's look for any files that start with the map. -map name*.* The matching files are listed. Note that they are all in a subdirectory. Searching for directories with finding the -path option makes it look like directories. Let's look for a directory that we can't quite remember the name, but we know it ends with the letters about. Find. -the path * about the directory is found, is just called about, and is nested inside another directory in the current directory. There is an -ipath option (ignore the case path) allows you to search for paths and ignore the case, similar to the -iname option discussed above. Using file attributes with finding can search for files that have attributes match the search clue. For example, you can search for empty files by using the -empty option, whatever they are called. Find. -empty Any zero byte length file will be listed in the search results. The -executable option will find any file that can be run, would be a program or a script. Find. -Executable Results lists a file called fix_aptget.sh. They also contain three directories, including ., the current director. Directories are included in the results because the execution bit is set in their file permissions. Without this, you wouldn't be able to change into (run) these directories. The -type option The -type option allows you to search for the type of object you are looking for. We will provide the f indicator as a parameter of the type option because we want to find only files. Find. executable -type f This time subdirectories are not listed. The executable script file is the only element in the results. We can also ask find find to include only directories in the results. To list all directories, we can use the type option with the type d indicator. type -d Only directories and subdirectories are listed in the results. Using other find able commands You can perform some additional actions on found files. You can have your files switched to another command. If we need to make sure that there are no executable files in the current directory and subdirectories, we could use the following command: find . -name fix_aptget.sh -exec chmod -x {} \; The command means: Search the current directory for an object named fix_aptget.sh. If found, run the chmod command. The parameters that are transmitted to chmod are -x to remove executable permissions and {} that represent the file name of the found file. The final semicolon marks the end of the parameters that will be passed to chmod. This must be gotten rid of it by preceding it with a backslash. Once this command has been executed, we can search for executable files as before and this time there will be no more files listed. To throw our net wider, we could use a file model instead of the file name we used in our example. This flexibility allows you to search for specified file types or file name patterns and perform some actions on matching files. Find has many other options, including searching for files by date of change, files owned by a user or group, files that can be read, or files that have a specific set of file permissions. Locating and mlocated commands many Linux distributions used to have a copy of the location included with them. This was replaced by the mlocated command, which was an improved and updated version of the When mlocate is installed on a system, it changes the location command so that you actually use mlocate if you type locate. Current versions of Ubuntu, Fedora, and Manjaro were checked to see if they had versions of these commands pre-installed on them. Ubuntu and Fedora both included mlocated. It had to be installed on Manjaro, with this command: sudo pacman -Syu mlocate On Ubuntu, you can use locate and interchangeable mlocate. On Fedora and Manjaro you must type locate, but the command is executed for you by mlocate. If you use the --version with locate option you will see that the responding command is actually mlocated. locate --version Because locate works on all Linux distributions that have been tested, we use localize in our explanations below. And it's less of a letter to type. Location of the database The biggest advantage that localization has is speed. When you use the find command, it stops and performs a search in the file system. The location command works very differently. It does a database search to determine if what you're looking for is on your computer. That makes the search a lot faster. Of course, it raises an obvious question about the database. What makes sure the database is up to date? When installed mlocate is (usually) places an entry in cron.daily. This runs every day (very early in the morning) and updates the database. To verify that this entry exists, use this command: ls /etc/cron.daily/*place* If you can't find an entry there, you can set up an automatic task to do so when you choose. RELATED by: to schedule tasks on Linux: An introduction to Crontab files What if your computer is not on at the time the database should be updated? You can manually run the database update process with the following command: sudo updatedb Use Location Let's search for files that contain the getlatlong string. With location, search automatically searches for all matches that contain the search term anywhere in the file name, so you don't need to use wildcard characters. locate getlatlong It's hard to transmit speed in a screenshot, but almost immediately the matching files are listed for us. Tell locate how many results you want Sometimes you may know that there are a lot of files of the type you are looking for. You just have to see the first ones. Maybe you just want to be reminded what directory I'm in, and you don't need to see all the file names. Using the -n (number) option, you can limit the number of results that will locate you. In this order, we set a limit of 10 results. locate .html -n 10 locate responds by listing the first 10 matching file names retrieved from the database. Count the right files If you want know the number of files that match and you don't need to know are named or where they are on your hard disk, use the -c (count) option. locate -c .html So now we know that there are 431 files with extension on this computer. We may want to look at them, but we thought we'd take a look and see how many were the first. Armed with this knowledge we know that we will need to pipe out through less. .html location | Less and here are all, or at least, here's the top of the long list of them. Ignoring the case with locate them (ignore the case) causes the locate to do just that, ignore the uppercase and lowercase differences between the search term and the file names in the database. If we try to count HTML files again, but accidentally offer the search term in capital letters, we will get zero results. locate -c . HTML By including the option we can make the locate ignore the difference in the case, and return our expected response to this device, which is 431. locate -c -i . HTML Location of database status To see the status of the database, use the -s (status) option. This causes the locate to return some statistics about the size and content of the database. Locate that command that command searches through the directories in your path, and tries to locate the command you're looking for. It allows you to determine which version of a program or command will run when you type its name on the command line. Imagine we have a program called a geoloc. We know it's installed on your computer, but we don't know where it is. He's got to be somewhere in the way because when we type his name, he runs. We can use that to locate it with this command: which geoloc that reports that the program is located in /usr/local/bin. We can check if there are other copies of the program in other locations within the path using the -a option (all). that a geoloc This shows us that we have the geoloc program in two places. Of course, the copy in /usr/local/bin will be found first by the Bash shell each time, so the program in two places is meaningless. Removing the version in /usr/bin/geoloc will save you a bit of hard disk capacity. More importantly, it will also avoid problems created by someone manually updating the program, and do it in the wrong place. Then wondering why I don't see the new updates when running the program. The whereis command Whereis command is similar to the which command, but it is more informative. In addition to the location of the command or file program, where it is also reports where the man (manual) pages and source code files are located. In most cases, the source code files will not be on your computer, but if they are, where will they be reported. Binary executable, human pages, and source code are often referred to as a package for that command. If you want to know where the different components of the diff command package are, use the following command: whereis diff respond by listing the location of diff man pages and binary file. To restrict results to display only the binary location (actually, make whereis work like which) use the -b (binary) option. where -b diff where it relates only to the location of the executable file. To restrict the search to report only on human pages use the option -m (manual). To restrict search to report only in source code files, use the -s (source) option. To see where to search, use the -l (locations) option. where it locations are listed for you. Now that we know where he'll be looking, we can, if we choose, restrict the search to a specific location or group of locations. The -B (binary list) option restricts the search for executable files to the command-line list of paths. You must provide at least one location for where to search through. The -f (file) option is used to flag the end of the location last time at the beginning of the file name. whereis -B/bin/-f chmod whereis show in the only place I asked to look through. It happens to be where the file is. You can also use the -M (manual list) option to restrict man's page searches to the paths you provide on the command line. The -S (source list) option allows you to restrict the search for source code files in the same way. The whatis command whatis command is used to quickly search through human (manual) pages. It provides summary descriptions on a single line of the term you asked him to look for. Let's start with a simple example. Although it seems to be the starting point of the profound philosophical debate, we only wonder what it means to tell us what the term man means. Whatis man whatis finds two matching descriptions. Prints a short description for each match. It also lists the numbered section of the manual that contains each full description. To open the manual in the section describing the man command, use the following command: man 1 man The manual opens in the man section(1) on the man page. To open the manual in section 7, to the page that discusses macros you can use to generate human pages, use this command: man 7 man page man for man macros is displayed for you. Searching in certain sections of the Option -s (section) manual is used to limit the search to the sections of the manual that interest you. To make the whatis search limited to section 7 of the manual, use the following command. Note the quotation marks around the section number: whatis -s 7 man Results refer only to section 7 of the manual. Using whatis With Wildcards You can use wildcard characters with whatis. You must use the -w (wildcard) option to do this whatis -w char* The right results are listed in the terminal window. Order by the way Order by the way is similar to what it is, but it still has a few bells and whistles. Search through the titles of the man's pages. Human. descriptions of a line that searches for the search term. It lists the descriptions of the right man's pages in the terminal window. The word apropos means related to or regarding, and by the way the command took his name from this. To search for anything related to group command, we can use this command: groups by the way list the results in the terminal window. Using more than one search term, you can use more than one search term in the command line. by the way will search pages of man that contain any of the search terms. apropos chhown chmod Results are listed as before. In this case, there is only one entry for each of the search terms. Using Exact Matches apropos will return pages of man containing the search term, even if the term is in the middle of another word. To return only the exact matches for the search term by the way, use the -e (exact) option. To illustrate this, we use apropos with grep as the search term. apropos grep There are many results returned for this, including many if the graft is embedded in another word, would be bzfgrep. Let's try again and use the -e (accurate) option. by the way-- it's grep We only have one result this time, for what we were actually looking for. Matching all search terms After I saw earlier, if you provide more than one search term apropos will search pages of man that contain either search term. We can change this behavior using the -a (and) option. This causes by the way to select only matches that have all the search hours in them. Let's try ordering without an option, so we can see what results apropos gives. by the way crontab cron results include man pages that match one or the other of the search terms. Now we use the option -o. apropos -a crontab cron This time the results are restricted to those that contain both search terms. However, more options All of these commands have several options - some of them many more options - and you are encouraged to read the human pages for the commands we discussed in this article. Here's a quick summary for each command: find: Provides a feature-rich search capability and granularity to search for files and directories. Locate: Provides a quick database-based search for programs and commands. Which: Searches for $PATH search for whereis executable files: Searches for $PATH searches for executable files, human pages, and source code files. whatis: Search for one-line man descriptions for matches with the search term. by the way: Look for the man's page with more fidelity than what it is, for matches with the term or terms. Looking for more information about the Linux terminal? Here are 37 commands you should know. RELATED BY: 37 important Linux commands you should know