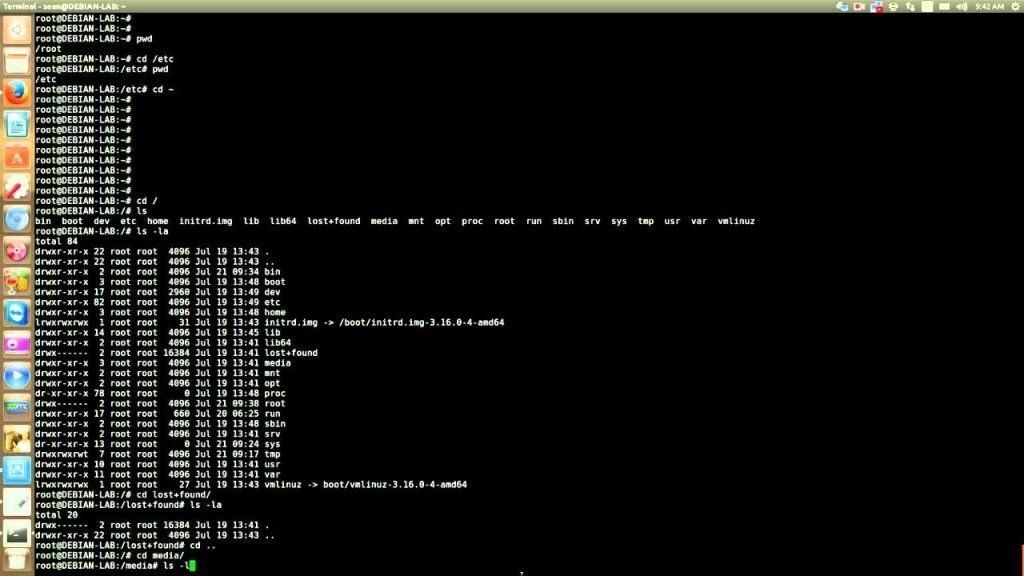
**Most Useful Linux Commands You Can Learn Today**

Below, you will find the 50 best Linux terminal commands our experts have handpicked in an aim to skyrocket your Linux system experience. By creatively implementing many of these flexible terminal commands into your next [terminal](https://www.ubuntupit.com/linux-terminal-emulator-top-15-reviewed-compared/) session, you will start to feel the heat of becoming a conscience Unix system user. Before going into the details, you can grab a set of [best Linux command cheat sheet](https://www.ubuntupit.com/top-40-best-linux-commands-cheat-sheet-get-it-free-now/) for mastering the Linux system.

**Linux Commands for Navigating the Filesystem**

As with every modern operating system, the Linux filesystem sits at the core of its fundamental design and paves the way for you to visualize and manipulate your system hierarchy. Learning to navigate the filesystem creatively will take your Linux skills to the next level.



**1. pwd**

pwd stands for Print Work directory and does exactly what you think – it shows the directory you’re currently in. This is one of the handiest Linux terminal commands that aims to make new user’s life peaceful by ensuring they don’t get lost in that seemingly cryptic terminal window.

**2. ls**

The ls command is probably one of the most widely used commands in the Unix world. It presents to you the contents of a particular directory – both files and directories. You will use this command alongside pwd to navigate your ways inside the mighty Unix filesystem.

Besides looking up your files, **ls** can help you examine your permissions. In the example below, you can't run myapp because of a permissions issue. When you check the permissions using **ls -l**, you realize that the permissions do not have an "x" in **-rw-r--r--**, which are read and write only.

**$ ./myapp  
bash: ./myapp: Permission denied  
$ ls -l myapp  
-rw-r--r--. 1 root root 33 Jul 21 18:36 myapp**

**3. cd**

Short for Change Directory, the cd command is behind your movement from one directory to another. It’s one of the few Linux commands that you’re *bound* to use throughout your stint with the Linux system. This command makes life in front of the terminal less scary for beginners while providing a standard method to browse the entire filesystem of your device.

**4. mkdir**

Want to create a new folder through the terminal? The mkdir command is created for just this specific purpose. It lets you create folders in anywhere you like in your Linux system – given you have got the necessary permission of course!

**5. rmdir**

The arch rival of the mkdir command, the rmdir command allows you to delete specific folders from your system without any hassles. Although many utilize the rm command for this purpose, screwing up parameters or even a single character with rm can do things you wouldn’t even dream. So, stick with rmdir for now.

**6. lsblk**

Often you will find the need to list the available block devices of your Linux system. The lsblk is one of the most used Linux commands for this purpose. This handy terminal command will present you with a tree structure of your block devices and is used heavily by professional users.

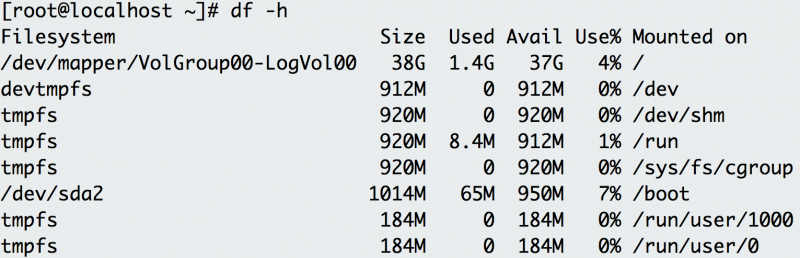
**7. mount**

Contrary to Windows, whenever you plug in an SD card or a USB, chances are your distro won’t show them directly at the start. You need to mount it with your existing filesystem using the mount command. This Linux command is one of the most powerful terminal commands out there.

**8. df**

The df command is one of the most empowering Linux terminal commands that display essential information about the disk space on your filesystem. It is used widely by system administrators to monitor and analyze real-time server or network oriented systems.

You can use **df** (display free disk space) to troubleshoot disk space issues. When you run your application on a container orchestrator, you might receive an error message signaling a lack of free space on the container host. While disk space should be managed and optimized by a sysadmin, you can use **df** to figure out the existing space in a directory and confirm if you are indeed out of space.

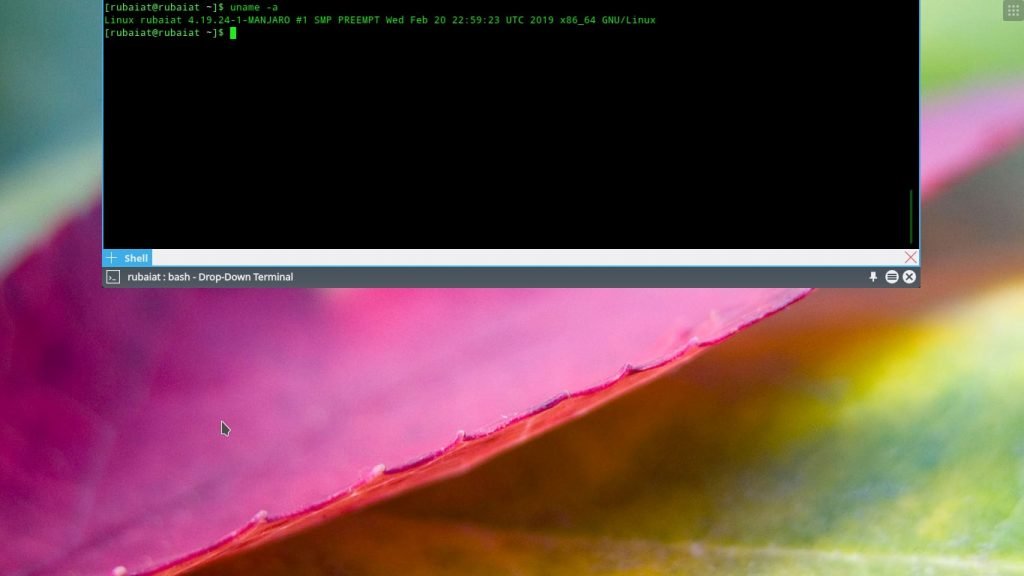


**Df**shows the disk space for each filesystem, its absolute space, and availability.

The **-h** option prints out the information in human-readable format. The example above shows plenty of disk space on this host.

**Linux Commands for Manipulating the System**

System commands are essential to gain information about your Linux system. A significant number of powerful Linux commands are at your disposal for this task. Below we’re presenting a few to get you started.



**9. uname**

The uname command is an elementary Linux command for obtaining system information like name, version and other system specific details. You can quickly check your OS and kernel version with this command and can know for sure the instruction length of your machine.

**10. ps**

This is among other straightforward terminal commands that will allow you to visualize what processes are currently run by your machine. It’s a very elegant yet flexible mean to analyze system resources and will enable you to manipulate system processes natively through the terminal. This command is considered as one of the basic and [best Linux monitoring tools](https://www.ubuntupit.com/most-comprehensive-list-of-linux-monitoring-tools-for-sysadmin/) available for Linux nerds.

**ps** shows process status. Use this command to determine a running application or confirm an expected process. For example, if you want to check for a running Tomcat web server, you use **ps** with its options to obtain the process ID of Tomcat.

**$ ps -ef  
UID        PID  PPID  C STIME TTY          TIME CMD  
root         1     0  2 18:55 ?        00:00:02 /docker-java-home/jre/bi  
root        59     0  0 18:55 pts/0    00:00:00 /bin/sh  
root        75    59  0 18:57 pts/0    00:00:00 ps -ef**

For even more legibility, use **ps** and pipe it to **grep**.

**$ ps -ef | grep tomcat  
root         1     0  1 18:55 ?        00:00:02 /docker-java-home/jre/bi**

**11. kill**

The kill command is a powerful way to stop processes that are stuck due to resource constraints. As you grow your Linux system skills, you will come to know the essence and importance of this command. Often presented in lists of funny Linux commands, the kill command is as mighty as its name.

**12. service**

The service command is the de-facto command to invoke system-wide services from the terminal. A powerful Linux terminal command for manipulating the system; you can leverage this command for running any *System V init* script directly from the terminal window.

**13. batch**

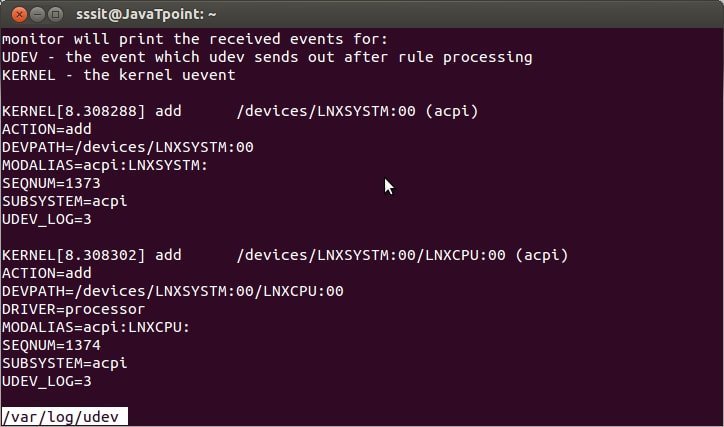
If you are looking for a neat tool that will run system services in a pre-defined schedule, the batch command is here for your redemption. Another powerful Linux command for writing automation shell scripts; this flexible little command can aid to your [Linux productivity](https://www.ubuntupit.com/top-15-linux-productivity-software-and-tools-that-worth-your-attention/) significantly.

**14. shutdown**

Feeling a bit lazy and don’t want to click a few GUI elements at the end of a long night? The shutdown command is here for empowering your Linux terminal commands skills to a whole new level. Just don’t type it now while reading this post!

**Linux Commands for Taking Care of Your Files**

Files sit at the heart of your Linux system. Almost everything you see on your system is some kind of a file and can be manipulated accordingly. Mastering the ways of [managing your files](https://www.ubuntupit.com/linux-file-manager-top-10-reviewed-for-linux-users/) is essential if you want to be a Linux guru.



**15. touch**

The touch command is an essential Linux command for creating a valid empty file. You can create files on the go in your terminal and fill them up later or in real-time – based on your requirements. It’s also the go-to command for changing the timestamps.

**16. cat**

Designed initially for concatenating multiple files, the cat command is used for numerous other purposes since. This is among other Linux commands you will use to create new files, view file contents in**cat** concatenates and prints files. You might issue **cat** to check the contents of your dependencies file or to confirm the version of the application that you have already built locally.

**$ cat requirements.txt  
flask  
flask\_pymongo**

The example above checks whether your Python Flask application has Flask listed as a dependency.

the terminal, and redirect output to another command line tool or file.

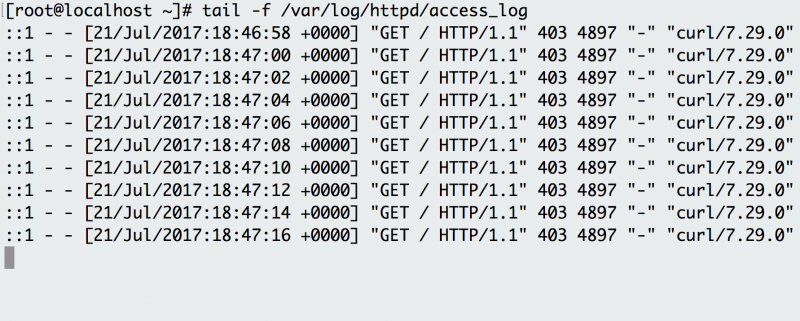
**17. head**

The head command allows you to view the beginning of a file or piped data directly from the terminal. It’s one of the most widely used Linux commands by users who works heavily with [text processing](https://www.ubuntupit.com/best-latex-editor-top-33-reviewed-for-linux-nerds/). Use this command whenever you are going through a lot of files in the terminal to increase your productivity.

**18. tail**

A compliment to the previous command, chances are you will use the tail command much more than the header commands. A basic Linux terminal command, tail, mixed with cat and echo can do things you wouldn’t even imagine.

**tail**displays the last part of a file. You usually don't need every log line to troubleshoot. Instead, you want to check what your logs say about the most recent request to your application. For example, you can use **tail** to check what happens in the logs when you make a request to your [Apache HTTP server](https://httpd.apache.org/).



Use**tail -f**to follow Apache HTTP logs and see the requests as they happen.

The **-f** option indicates the "follow" option, which outputs the log lines as they are written to the file. The example has a background script that accesses the endpoint every few seconds and the log records the request. Instead of following the log in real time, you can also use **tail** to see the last 100 lines of the file with the **-n** option.

**$ tail -n 100 /var/log/httpd/access\_log**

**19. cp**

The cp command is just a short way of telling your machine to copy a file or directory from one folder to another. It is among other de-facto Linux commands you can’t live without. You can copy multiple files to a directory right from your terminal with this neat command.

**20. mv**

Short for move, it’s a supplement to the cut operation you perform in the GUI. Just like cp, you can use the mv command to move either single or multiple files from one location to another. You can force this Linux command to transfer large files with the -f parameter.

**21. comm**

One of the original Unix commands that found its way to the Linux world, you can use comm to compare two files for common and distinct lines. This is an essential tool for many that need to process large amounts of files on the terminal.

**22. less**

Another most used Linux command, the less command is widely popular due to the convenience it brings when viewing the contents of a file. Contrary to cat, the less command allows users to navigate within a file in both directions while not interrupting the terminal session.

**23. ln**

The ln command is one of the handiest Linux commands for creating symbolic links to some specific file. You can use this tiny yet flexible command to produce multiple instances of a symbolic link to a particular file or directory on your disk space.

**24. cmp**

If you want to compare two files and print the result to the standard output stream, the cmp command will let you do exactly so. This, alongside the comm command, is one of the most used Linux commands by users that process large volumes of text files on a regular basis.

**25. dd**

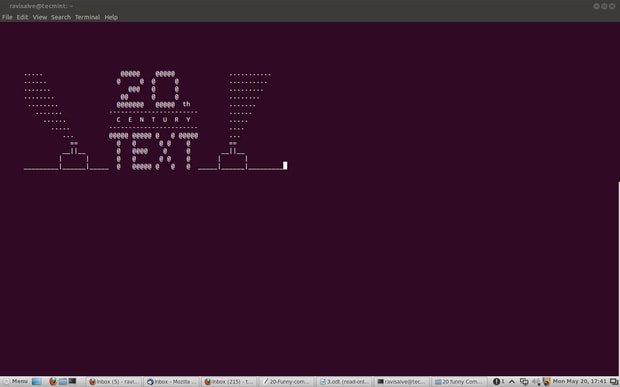
This is one of the most used Linux commands by veteran users to copy and convert files from one type to another. The exciting thing about this little yet powerful command is that you will often use it among other terminal commands when creating bootable live USB sticks.

**26. alias**

It is one of the most used Linux commands by system admins as it lets them replace a word by another string in files directly from the terminal. Among other functions, this is one of the best terminal commands through which you can customize the shell and manipulate the environment variables.

**Funny Linux Commands to Explore When Bored**

A lot of funny Linux commands are available to make your time in front of the terminal exciting. These terminal commands are funny in a way that they do things no one would imagine them to do. We highly encourage you to play with such funny Linux commands as they’ll redeem your boredom while providing deep insights into the Unix philosophy.



**27. cal**

Did you know Unix provides a handy little calendar tool right into the terminal? The cal command is one of those funny Linux commands that shows you the calendar in an ASCII text format. Type this command with parameters like month and year to get specified information right into the terminal.

**28. fortune**

This is one of my most favorite funny Linux commands. Just type it in the terminal and see for yourself! It shows out of the blue poignant, inspirational, or silly phrases that will definitely make you laugh out loud.

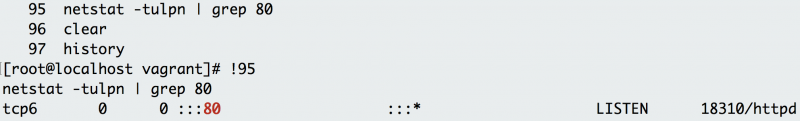
**29. history**

Want to check the history of your terminal sessions? The history command lets you do precisely this. When typed without any parameters, this will print out the bash history of your terminal session right in the tiny window. It’s not only among the most useful funny Linux commands you will use in the terminal but also provides a great insight into your terminal usage.

When you issue so many commands for testing and debugging, you may forget the useful ones! Every shell has a variant of the **history** command. It shows the history of commands you have issued since the start of the session. You can use **history** to log which commands you used to troubleshoot your application. For example, when you issue **history** over the course of this article, it shows the various commands you experimented with and learned.

**$ history  
    1  clear  
    2  df -h  
    3  du**

What if you want to execute a command in your previous history, but you don't want to retype it? Use **!** before the command number to re-execute.



Adding**!**before the command number you want to execute issues the command again.

**30. yes**

Okay, you want some real funny Linux commands to try right now? Just open up a terminal type in yes then some texts of your choosing and press enter. We’re pretty sure it’s among those terminal commands you will never forget ever since.

**31. banner**

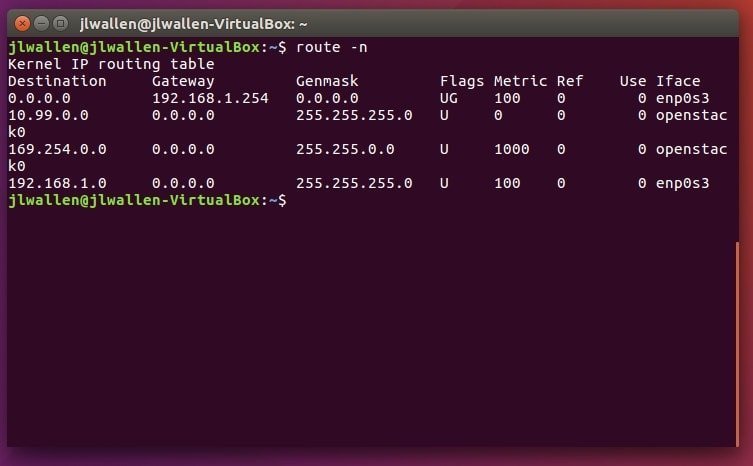
Ever saw some great ASCII banners used by old school Unix legends and fall in love right away? The banner command is one of those funny Linux commands that will empower you to create your own personalized such banners. Just enter this command mixed with some texts in the terminal to get a feel.

**32. rev**

This is another funny terminal command for veteran Linux users. The rev command will take your input text and write them to the standard output, reversing each character! It’s one of those terminal commands you can use to make a cryptic yet subtle impression on your friends.

**Most Used Linux Commands for Network Administrators**

It is the power and flexibility Linux provides for networking that makes it the go-to solution for sysadmins worldwide. Linux is behind most of the computer networks that we’re familiar with today. As this guide is meant for beginners, we’re outlining only the most elementary terminal commands for networking.



**33. wget**

This is one of the best Linux commands network admins leverage to [download files](https://www.ubuntupit.com/download-manager-for-linux-top-8-reviewed-compared/) from the web right from the terminal. This is among those handy little terminal commands that can be used in scripts or cronjobs and provides users the ability to use the HTTP, HTTPS, and FTP internet protocol.

**34. iptables**

The [iptables](https://www.ubuntupit.com/top-10-linux-firewall-software-for-protecting-your-linux-systems#1-iptables) command invokes a terminal utility that lets system admins control the incoming and outgoing internet traffic on a particular host machine. It is among the most used Linux commands sysadmins use on a regular basis to define authentic traffics and for blacklisting suspicious or untrusted network requests.

**iptables** blocks or allows traffic on a Linux host, similar to a network firewall. This tool may prevent certain applications from receiving or transmitting requests. More specifically, if your application has difficulty reaching another endpoint, **iptables** may be denying traffic to the endpoint. For example, imagine your application's host cannot reach [Opensource.com](https://opensource.com/). You use **curl** to test the connection.

**$ curl -vvv opensource.com  
\* About to connect() to opensource.com port 80 (*#0)*  
\*   Trying 54.204.39.132...  
\* Connection timed out  
\* Failed connect to opensource.com:80; Connection timed out  
\* Closing connection 0  
curl: (7) Failed connect to opensource.com:80; Connection timed out**

The connection times out. You suspect that something might be blocking the traffic, so you show the **iptables** rules with the **-S** option.

**$ iptables -S  
-P INPUT DROP  
-P FORWARD DROP  
-P OUTPUT DROP  
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT  
-A INPUT -i eth0 -p udp -m udp --sport 53 -j ACCEPT  
-A OUTPUT -p tcp -m tcp --sport 22 -j ACCEPT  
-A OUTPUT -o eth0 -p udp -m udp --dport 53 -j ACCEPT**

The first three rules show that traffic drops by default. The remaining rules allow SSH and DNS traffic. In this case, follow up with your sysadmin if you require a rule to allow traffic to external endpoints. If this is a host you use for local development or testing, you can use the **iptables** command to allow the correct traffic. Use caution when adding rules that allow traffic to your host.

**35. traceroute**

This command is widely used by security professionals who leverage this command with other terminal commands for determining the route a network packet takes on its way from one machine to another. This is a compelling network command by using which you can safeguard your computer from a number of harmful intruders.

**36. cURL**

cURL is a very powerful network tool that makes transferring files over a network a child’s play for even new Linux system users. This is one of those Linux commands designed to work without user interaction and is typically employed in network related shell scripts.

**curl** transfers a URL. Use this command to test an application's endpoint or connectivity to an upstream service endpoint. **curl** can be useful for determining if your application can reach another service, such as a database, or checking if your service is healthy.

As an example, imagine your application throws an HTTP 500 error indicating it can't reach a MongoDB database:

**$ curl -I -s myapplication:5000  
HTTP/1.0 500 INTERNAL SERVER ERROR**

The **-I** option shows the header information and the **-s** option silences the response body. Checking the endpoint of your database from your local desktop:

**$ curl -I -s database:27017  
HTTP/1.0 200 OK**

So what could be the problem? Check if your application can get to other places besides the database from the application host:

**$ curl -I -s https://opensource.com  
HTTP/1.1 200 OK**

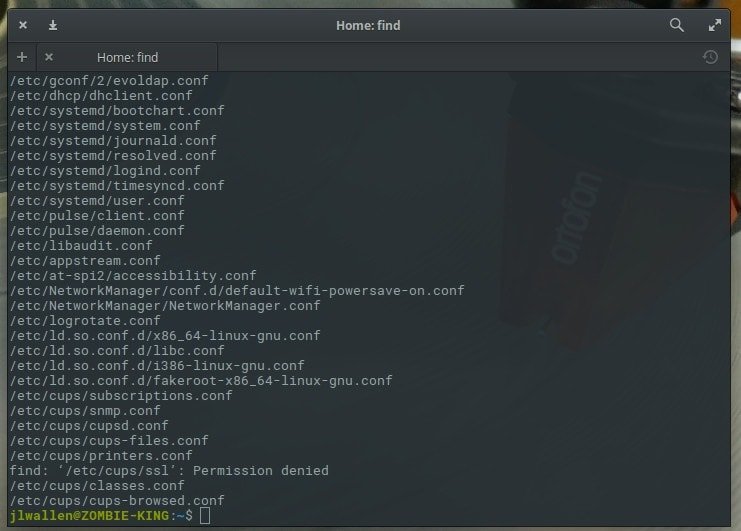
That seems to be okay. Now try to reach the database from the application host. Your application is using the database's hostname, so try that first:

**$ curl database:27017  
curl: (6) Couldn't resolve host 'database'**

This indicates that your application cannot resolve the database because the URL of the database is unavailable or the host (container or VM) does not have a nameserver it can use to resolve the hostname.

**Linux Commands for Search and Regular Expression**

Linux provides a handful of flexible terminal commands for searching the machine effectively. You can combine such Linux terminal commands with powerful regular expression commands to gain information regarding a particular file or sequence in files pretty quickly.



**37. find**

The find command is one of the most used Linux commands to search for files from the terminal. This compelling yet flexible terminal command allows users to search for files based on certain criteria such as file permissions, ownership, modification date, size, etc.

**38. which**

The which command is pretty useful if all you are trying to search are executable files. This handy little terminal command takes specific parameters and searches for binary files in the $PATH system environment variable based on them very effectively.

**39. locate**

The locate command is one of those Linux commands that are used for finding the location of a specific file. It is one of the most straightforward terminal commands that you can leverage when not sure about the location of a particular file on your Linux machine.

**40. grep**

This is among the most powerful regular expression terminal commands you can use when searching for patterns inside large volumes of text files. It will take the pattern you’re looking for as input and search the specified files for that particular pattern.\

**grep** searches file patterns. If you are looking for a specific pattern in the output of another command, **grep** highlights the relevant lines. Use this command for searching log files, specific processes, and more. If you want to see if Apache Tomcat starts up, you might become overwhelmed by the number of lines. By piping that output to the **grep** command, you isolate the lines that indicate server startup.

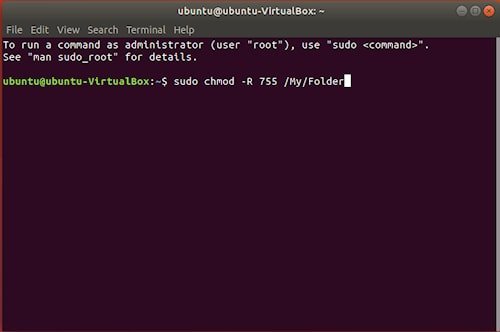
**$ cat tomcat.log | grep org.apache.catalina.startup.Catalina.start  
01-Jul-2017 18:03:47.542 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 681 ms**

**41. sed**

This is one of the most widely used Linux commands to manipulate each line of a file or stream by replacing specified parts. It is used heavily by users that deal with large volumes of text data and needs to change them on the go.

**Linux Commands That Deal With I/O And Ownership**

Linux offers a robust set of terminal commands to set and manipulate the I/O streams and file or directory ownership. The Linux commands listed below will outline some of the most basic Linux terminal commands for such purposes.



**42. clear**

The clear command is handy to clear out your existing terminal screen. Often you will find the need to wipe out the terminal screen after some earlier Linux commands leave your terminal screen with a garbled output.

**43. echo**

The echo command is a very powerful command line utility that lets you output a specific text to the [terminal console](https://www.ubuntupit.com/top-15-online-linux-terminal-emulators-and-bash-editors/). Type in echo followed by some texts within parentheses to find out for yourself. What’s more interesting for this command is that you can pipe the output to other terminal commands.

**44. sort**

The sort command is quite compelling at the things it does. Whenever you find the need to sort out a file in an alphabetical or reverse manner, utilize this command.

**45. sudo**

The sudo command is the holy grail of Linux commands. It lets non-privileged users access and modify files that require low-level permissions. Often you will use this command to access root from your regular user account.

**46. chmod**

The chmod command is among the most powerful Linux commands you will use to change or modify the access permissions of system files or objects. This command can take a very diverse set of parameters from users and based on those change the file permission.

When you run your application binary for the first time on your host, you may receive the error message "permission denied." As seen in the example for **ls**, you can check the permissions of your application binary.

**$ ls -l  
total 4  
-rw-rw-r--. 1 vagrant vagrant 34 Jul 11 02:17 test.sh**

This shows that you don't have execution rights (no "x") to run the binary. **chmod** can correct the permissions to enable your user to run the binary.

**$ chmod +x test.sh  
[vagrant@localhost ~]$ ls -l  
total 4  
-rwxrwxr-x. 1 vagrant vagrant 34 Jul 11 02:17 test.sh**

As demonstrated in the example, this updates the permissions with execution rights. Now when you try to execute your binary, the application doesn't throw a permission-denied error. **Chmod** may be useful when you load a binary into a container as well. It ensures that your container has the correct permissions to execute your binary.

**47. chown**

The chown command is very much similar to the chmod command. But instead of changing access permissions, it enables users to change the ownership of a file or directory. Both the chmod and chown terminal commands require root privileges to run.

**Miscellaneous Linux Commands For Everyday Use**

The below terminal commands will help you increase productivity and decrease your workload. Use these commands whenever you see them dim fit for the context.

**48. man**

The man command stands for manual and is one of the most useful Linux commands you can get your hands on. This command followed by the name of another command lists the manual or documentation page of that command. You will need to use this command often when determining how to use specific terminal commands.

**49. tar**

The tar command is used for archiving files and extracting them. It is a very popular command for compressing files and can tackle such tasks very efficiently.

**50. whatis**

The whatis command traverses a set of databases with short descriptions provided by the user and prints out system commands that match them.

**51. env**

**env** allows you to set or print the environment variables. During troubleshooting, you may find it useful for checking if the wrong environment variable prevents your application from starting. In the example below, this command is used to check the environment variables set on your application's host.

**$ env  
PYTHON\_PIP\_VERSION=9.0.1  
HOME=/root  
DB\_NAME=test  
PATH=/usr/local/bin:/usr/local/sbin  
LANG=C.UTF-8  
PYTHON\_VERSION=3.4.6  
PWD=/  
DB\_URI=mongodb://database:27017/test**

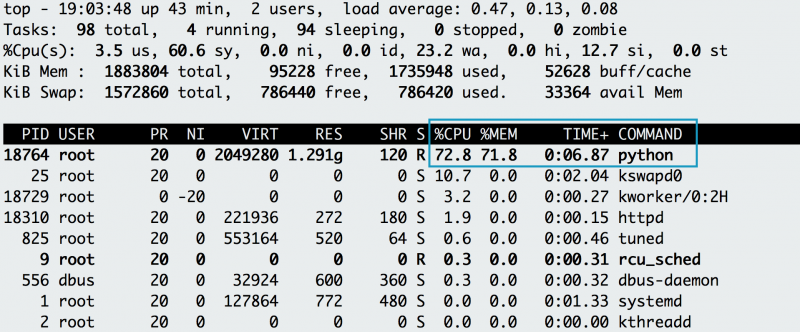
Notice that the application is using Python3 and has environment variables to connect to a MongoDB database.

**52. top**

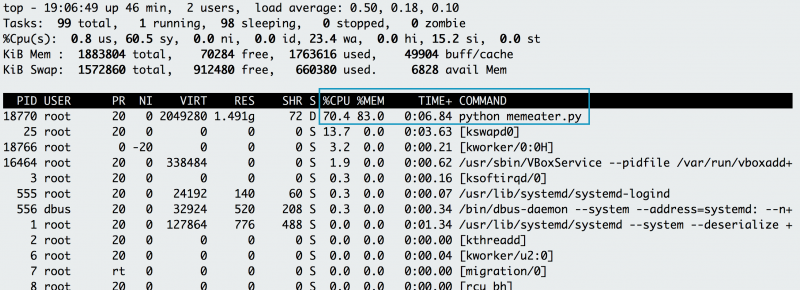
**top** displays and updates sorted process information. Use this tool to determine which processes are running and how much memory and CPU they consume. A common case occurs when you run an application and it dies a minute later. First, you check the application's return error, which is a memory error.

**$ tail myapp.log  
Traceback (most recent call last):  
MemoryError**

Is your application*really*out of memory? To confirm, use **top** to determine how much CPU and memory your application consumes. When issuing **top**, you notice a Python application using most of the CPU, with its memory usage climbing, and suspect it is your application. While it runs, you hit the "C" key to see the full command and reverse-engineer if the process is your application. It turns out to be your memory-intensive application (**memeater.py**). When your application has run out of memory, the system kills it with an out-of-memory (OOM) error.



The memory and CPU usage of the application increases, eventually being OOM-killed.

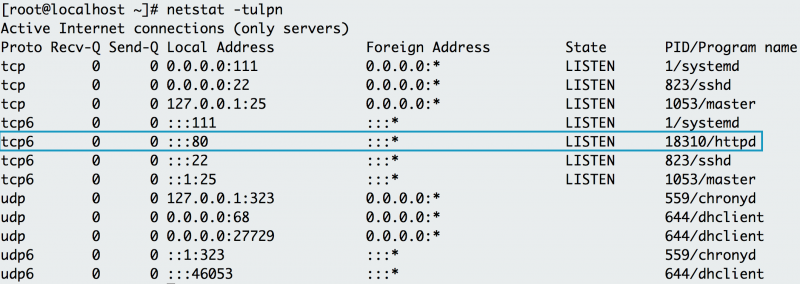


By hitting the "C" key, you can see the full command that started the application.

In addition to checking your own application, you can use **top** to debug other processes that utilize CPU or memory.

**53. netstat**

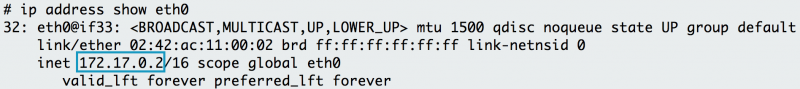
**netstat** shows the network status. This command shows network ports in use and their incoming connections. However, **netstat** does not come out-of-the-box on Linux. If you need to install it, you can find it in the [**net-tools**](https://wiki.linuxfoundation.org/networking/net-tools) package. As a developer who experiments locally or pushes an application to a host, you may receive an error that a port is already allocated or an address is already in use. Using **netstat** with protocol, process and port options demonstrates that Apache HTTP server already uses port 80 on the below host.



Using**netstat -tulpn**shows that Apache already uses port 80 on this machine.

**54. ip address**

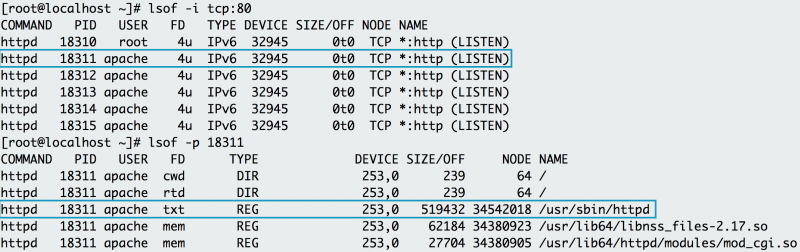
If **ip address** does not work on your host, it must be installed with the [**iproute2**](https://wiki.linuxfoundation.org/networking/iproute2) package. **ip address** shows the interfaces and IP addresses of your application's host. You use **ip address** to verify your container or host's IP address. For example, when your container is attached to two networks, **ip address** can show which interface connects to which network. For a simple check, you can always use the **ip address** command to get the IP address of the host. The example below shows that the web tier container has an IP address of 172.17.0.2 on interface eth0.



Using**ip address**shows that the IP address of the eth0 interface is 172.17.0.2

**55. lsof**

**lsof** lists the open files associated with your application. On some Linux machine images, you need to install **lsof** with the [lsof](http://www.linuxfromscratch.org/blfs/view/svn/general/lsof.html" \t "_blank) package. In Linux, almost any interaction with the system is treated like a file. As a result, if your application writes to a file or opens a network connection, **lsof** will reflect that interaction as a file. Similar to **netstat**, you can use **lsof** to check for listening ports. For example, if you want to check if port 80 is in use, you use **lsof** to check which process is using it. Below, you can see that httpd (Apache) listens on port 80. You can also use **lsof** to check the process ID of httpd, examining where the web server's binary resides (**/usr/sbin/httpd**).



**Lsof**shows that httpd listens on port 80. Examining httpd's process ID also shows all the files httpd needs in order to run.

The name of the open file in the list of open files helps pinpoint the origin of the process, specifically Apache.

**56. du**

To retrieve more detailed information about which files use the disk space in a directory, you can use the **du** command. If you wanted to find out which log takes up the most space in the **/var/log** directory, for example, you can use **du** with the **-h** (human-readable) option and the **-s** option for the total size.

**$ du -sh /var/log/\*  
1.8M  /var/log/anaconda  
384K  /var/log/audit  
4.0K  /var/log/boot.log  
0 /var/log/chrony  
4.0K  /var/log/cron  
4.0K  /var/log/maillog  
64K /var/log/messages**

The example above reveals the largest directory under **/var/log**to be **/var/log/audit**. You can use **du** in conjunction with **df** to determine what utilizes the disk space on your application's host.

**57. id**

To check the user running the application, use the **id** command to return the user identity. The example below uses [Vagrant](https://www.vagrantup.com/) to test the application and isolate its development environment. After you log into the Vagrant box, if you try to install Apache HTTP Server (a dependency) the system states that you cannot perform the command as root. To check your user and group, issue the **id** command and notice that you are running as the "vagrant" user in the "vagrant" group.

**$ yum -y install httpd  
Loaded plugins: fastestmirror  
You need to be root to perform this command.  
$ id  
uid=1000(vagrant) gid=1000(vagrant) groups=1000(vagrant) context=unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023**

To correct this, you must run the command as a superuser, which provides elevated privileges.

**58. dig / nslookup**

A domain name server (DNS) helps resolve a URL to a set of application servers. However, you may find that a URL does not resolve, which causes a connectivity issue for your application. For example, say you attempt to access your database at the **mydatabase** URL from your application's host. Instead, you receive a "cannot resolve" error. To troubleshoot, you try using **dig** (DNS lookup utility) or **nslookup** (query Internet name servers) to figure out why the application can't seem to resolve the database.

**$ nslookup mydatabase  
Server:   10.0.2.3  
Address:  10.0.2.3*#53*  
  
\*\* server can't find mydatabase: NXDOMAIN**

Using **nslookup** shows that **mydatabase** can't be resolved. Trying to resolve with **dig** yields the same result.

**$ dig mydatabase  
  
; <<>> DiG 9.9.4-RedHat-9.9.4-50.el7\_3.1 <<>> mydatabase  
;; global options: +cmd  
;; connection timed out; no servers could be reached**

These errors could be caused by many different issues. If you can't debug the root cause, reach out to your sysadmin for more investigation. For local testing, this issue may indicate that your host's nameservers aren't configured appropriately. To use these commands, you will need to install the [**BIND Utilities**](http://www.linuxfromscratch.org/blfs/view/svn/basicnet/bind-utils.html) package.

**59. sestatus**

You usually find SELinux (a Linux security module) enforced on an application host managed by an enterprise. SELinux provides least-privilege access to processes running on the host, preventing potentially malicious processes from accessing important files on the system. In some situations, an application needs to access a specific file but may throw an error. To check if SELinux blocks the application, use **tail** and **grep** to look for a "denied" message in the **/var/log/audit** logging. Otherwise, you can check to see if the box has SELinux enabled by using **sestatus**.

**$ sestatus  
SELinux status:                 enabled  
SELinuxfs mount:                /sys/fs/selinux  
SELinux root directory:         /etc/selinux  
Loaded policy name:             targeted  
Current mode:                   enforcing  
Mode from config file:          enforcing  
Policy MLS status:              enabled  
Policy deny\_unknown status:     allowed  
Max kernel policy version:      28**

The output above indicates that the application's host has SELinux enabled. On your local development environment, you can update SELinux to be more permissive. If you need help with a remote host, your sysadmin can help you determine the best practice for allowing your application to access the file it needs.