**10 basic Linux commands you need to know**

Out of the hundreds of possible Linux commands you could learn, here are the ten basic ones you really need to know.



Whether you're a Windows system administrator looking to expand your skills into Linux, a fresh convert to Linux, or someone who's looking to find a job in IT, this introduction to some common Linux commands is for you. These 10 commands are the ones that Linux system administrators use frequently—in fact, they use at least eight of them on a daily basis. And no matter how long you use Linux, you'll always use these commands. Some of you MS-DOS users will recognize a few of these and, not surprisingly, they have the same function in both operating systems.

## ls lists directory contents

## More Linux resources

* [Download Now: Linux Commands Cheat Sheet](https://developers.redhat.com/promotions/linux-cheatsheet/?intcmp=701f20000012ngPAAQ)
* [Advanced Linux Commands Cheat Sheet for Developers](https://developers.redhat.com/cheat-sheets/advanced-linux-commands/?intcmp=701f20000012ngPAAQ)
* [Download Red Hat Enterprise Linux Server 8 Trial](https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux/try-it?intcmp=701f20000012ngPAAQ)
* [Linux System Administration Skills Assessment](https://www.redhat.com/rhtapps/assessment/session/?intcmp=701f20000012ngPAAQ)

The list (ls) command is equivalent to the DOS DIR command, in that it lists files and directories. If you simply type ls at a prompt ($), you'll see all non-hidden files in your current directory, which is your home directory when you first log into a Linux system. The ls command won't show you much in your home directory on a new system, so let's explore a directory that contains a lot of files and directories: /etc. The /etc (et-see) directory is where a Linux system's configuration files live.

$ ls /etc

A large number of files (over 200) appear on your screen. You've successfully listed the contents of the /etc directory, but you can actually list files in several different ways. Above, I mentioned non-hidden files. In your home directory, where you are now, you probably have hidden files. Hidden files in Linux begin with a period (.). For example, you likely have a .bash\_profile file there. To see it, use the following ls command.

$ ls -a

You now see several files beginning with a period. The -a switch—or option, as it's called—shows you all files, even hidden ones.

man displays manual pages

Linux has an extensive set of online documentation for your reference. They're referred to as manual pages, as in read the manual. The abbreviated command for referencing this documentation is, man <command> and a screen-full of information appears before you.

It's easy to navigate man pages. Use the **Enter** key to advance one line at a time, the 'b' key to go back, the **Space** bar to advance a full-screen page, and the 'q' key to exit the man page. As an example, look at the man page for the ls command.

$ man ls

cat concatenates files

The cat command is important as a basic command because it serves two very important functions: concatenating (merging) files (as the name suggests) and printing the contents of a file to the screen. Printing the contents of files is by far the more frequent use of this command. If you want to see a file's contents, use the following format:

$ cat <filename>

For example, you might type the following to display the contents of the system's passwd file on the screen:

$ cat /etc/passwd

To use cat for its file concatenation powers, the general form of the command is:

$ cat file1 file2 > file1file2

For example, to redirect the contents of grocerylist.txt and todo\_list.txt into the Saturday.txt file:

$ cat grocerylist.txt todo\_list.txt > Saturday.txt

You can concatenate as many files as you want into a single file using cat.

touch changes file timestamps

The touch command is another one that serves a dual purpose. Its designated purpose is to update the timestamps on files. If you list the contents of a directory in long format with:

$ ls -l

The command's output displays the permissions, ownership, size, created or last accessed date/time, and the filename:

-rw-rw-r--. 1 khess khess     175 Jul 23 19:19 all.txt  
-rw-rw-r--. 1 khess khess      61 Jul 23 19:17 new.txt  
-rw-rw-r--. 1 khess khess     114 Jul 23 19:09 students.txt

Use touch to update the last accessed timestamp:

$ touch new.txt

$ ls -l

-rw-rw-r--. 1 khess khess     175 Jul 23 19:19 all.txt  
-rw-rw-r--. 1 khess khess      61 Jul 26 11:28 new.txt  
-rw-rw-r--. 1 khess khess     114 Jul 23 19:09 students.txt

Using touch to update last accessed time is actually an infrequent use of this command. The common use for touch is to create an empty file as a placeholder. Some programs require that a file exists to operate correctly, and this is one method of kickstarting such a process. Otherwise, this use offers a quick way to create a file without opening a text editor and then saving an empty file:

$ touch today.txt

$ ls -l

-rw-rw-r--. 1 khess khess     175 Jul 23 19:19 all.txt  
-rw-rw-r--. 1 khess khess      61 Jul 26 11:28 new.txt  
-rw-rw-r--. 1 khess khess     114 Jul 23 19:09 students.txt  
-rw-rw-r--. 1 khess khess       0 Jul 26 11:53 today.txt

You have created a new empty file, today.txt.

pwd prints the working directory

The pwd command is your Linux system's compass, in that it tells you where you are. It has no other function than supplying that bit of information to you. Try the following, and you will see that you're in your home directory, which is shown in the format /home/<username>:  
$ pwd  
/home/khess

If you get lost, or just wonder where you are in the filesystem, this is the command that will tell you. Linux users use it frequently before changing or removing files to be sure of their current location.

The pwd command always displays the full path to your location, even if you're multiple directories deep from the root (/) directory, which is why I see /home/khess rather than khess or /khess.

cd changes directory

Very closely related to the pwd command is the cd command. Changing directories is a frequent activity on a Linux system. As stated before, when you first log in, you're placed into your home directory. Every user on a Linux system has a home directory. Regular user accounts have personal directories under the /home directory. Your home directory is under /home/<username>. To view all user's home directories, cd to the /home directory.

$ cd /home

$ ls

What you see here depends on your system. If you are the only user on a personal system, you will only see your home directory. Production systems might have hundreds of user accounts on them. The quick way to return to your home directory, no matter where you are on the system, is to type cd with no arguments or directory paths:

$ cd

So, if you ever get "lost" on the system and need to reset your bearings, type cd and you'll be placed safely into your home directory. You can cd to almost any directory on the system by supplying its full path after the cd command:

$ cd /usr/bin

To change directory to the one above your current directory, use the double period (dot) argument:

$ cd ..

Now you are in the /usr directory. Remember that you can "prove" your location to yourself by issuing the pwd command:

$ pwd  
/usr

There are times when you don't need to cd to a particular directory. You can read a file from your current location if you supply the full path to the file you're interested in viewing. For example, you don't need to cd to the /usr/bin directory to issue the pwd command. You issue it from your current location because it is in your path.

The path is a more advanced topic for another article, but just be aware that you don't need to cd to do everything. The time to cd is when you will be working in a specific directory for some reason. Otherwise, you can do what you need to do from your home directory. You'll find out why changing directories can be a bad thing in the next section.

rm removes files and directories

The rm command removes (deletes) files and directories. One of the quirks of Linux that you'll find different from DOS/Windows is that it isn't chatty, which means that when you remove a file or directory, you won't (by default) receive a message such as, "Are you sure?" It just isn't the Linux way. There is a recommended workaround for that behavior that I'll show you later in this section.

For now, let's remove the today.txt file that you created earlier with the touch command:

$ rm today.txt

Did you notice that you didn't receive any questions or prompts? Linux assumes you know what you want to do before you hit the **Enter** key. That's a little disconcerting, isn't it? Ask Linux system administrators if any files have ever gone missing during one of their sessions. I'll put money on an affirmative response and I'm not a gambler. You can work around this non-interactive behavior of certain commands by placing a -i switch (option) after the command. Try the following example:

$ touch newfile.txt

$ rm -i newfile.txt  
rm: remove regular empty file 'newfile.txt'?

The -i makes rm interactive. Answer with a y and the file goes away. Answer with an n and you keep the file. To be safe, you can always use the -i switch with rm. There's no harm or shame in it and you'll be glad you did at some point in the future.

cp copies files and directories

Copying files and directories is a very common task for Linux system administrators. There's no great secret to its usage and you simply issue the copy (cp) command, the file or directory source, and the destination. To copy a file, file.txt, to the /opt/files directory, use:

$ cp file.txt /opt/files

To copy an entire directory and all its contents, including subdirectories, use the -R (Recurse) option. Copy the data directory in your home directory to /opt/files. You can use either the -r or -R to recurse copy files:

$ cp -R data /opt/files

The cp command is rare in that both the upper- and the lowercase options for an action are the same. Of course, you can use wildcards when copying files to filter them with patterns:

$ cp \*.txt /opt/files

mkdir makes directories

If you're an organized person, you'll want to create directories to satisfy your need to properly arrange your files and data into separate compartments (directories). It's easy to create directories. Issue the mkdir command followed by the directory name you wish to create:

$ mkdir data

If you're even more organized and you've done some planning, you can create a whole hierarchy of directories with one command. You want to create a data directory that includes subdirectories for documents, forms, tests, and outgoing. Why issue multiple commands when you can do it all at once:

$ mkdir -p data/documents/forms/tests/outgoing

The -p option tells the system that you are creating a parent directory and subdirectories. Check your work using the ls command. You can also create multiple directories at the same level all at once.

$ mkdir docs spreadsheets email old

Use the ls command to be sure the mkdir command did what you wanted it to do.

ps lists the current running processes

The last of the 10 basic Linux commands you need to know is ps. This command shows you currently running processes. If you issue the ps command, you will only see your own processes:

$ ps  
PID TTY          TIME CMD  
7505 pts/0    00:00:00 bash  
18119 pts/0    00:00:00 ps

If you're not running anything, then this output is not very interesting. It's far more interesting to see what's going on system-wide. You can do this by adding some options to ps. The most valuable options are -e and -f, for every (all) and full format, respectively. To get the most information from the ps command, combine the two options into the following command. I've included the first few lines from the output of ps -ef from my system for you:

$ ps -ef

UID        PID  PPID  C STIME TTY          TIME CMD  
root         1     0  0 Jul23 ?        00:00:25 /usr/lib/systemd/systemd --system --deserialize 20  
root         2     0  0 Jul23 ?        00:00:00 [kthreadd]  
root         3     2  0 Jul23 ?        00:00:00 [rcu\_gp]  
root         4     2  0 Jul23 ?        00:00:00 [rcu\_par\_gp]  
root         6     2  0 Jul23 ?        00:00:00 [kworker/0:0H-kblockd]  
root         8     2  0 Jul23 ?        00:00:00 [mm\_percpu\_wq]

The fields are simple to understand and useful when troubleshooting performance problems:

| **Field** | **Description** |
| --- | --- |
| **C** | **CPU Usage.** |
| **CMD** | **The command or process name with path.** |
| **PID** | **Process ID.** |
| **PPID** | **Parent Process ID: The parent process is the one that spawned the process.** |
| **STIME** | **Start Time for the process.** |
| **TIME** | **CPU Time for the process.** |
| **TTY** | **The user terminal that spawned the process. System process will show a ?.** |
| **UID** | **User ID of the process owner.** |

There are other options you can use with the ps command, and it seems everyone has a preference, but the two most popular are: ps -ef and ps aux. They both provide you with a lot of process information.

There you have the 10 basic Linux commands you need to know. There isn't one command that's more important than any other. They're all important and they're all useful. I chose these because they are the 10 commands that everyone regularly uses whether you're a fresh newbie or an old salty system administrator from the dark days before Linux.

If you're interested in how DOS and Linux commands compare, check out this:

[Comparison of Common DOS and Linux Commands](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html).

# Appendix D. A Comparison of Common DOS and Linux Commands

Many Linux commands typed at a shell prompt are similar to the commands you would type in DOS. In fact, some commands are identical.

This appendix provides common commands used at the DOS prompt in Windows and their counterparts in Linux. Basic examples of how the command are used at the Linux shell prompt are also provided. Note that these commands usually have a number of options. To learn more about each command, read its associated man page (for example, type man ls at the shell prompt to read about the ls command).

| **Command's Purpose** | **MS-DOS** | **Linux** | **Basic Linux Example** |
| --- | --- | --- | --- |
| Copies files | copy | cp | cp *thisfile.txt* /home/*thisdirectory* |
| Moves files | move | mv | mv *thisfile.txt* /home/*thisdirectory* |
| Lists files | dir | ls | ls |
| Clears screen | cls | clear | clear |
| Closes shell prompt | exit | exit | exit |
| Displays or sets date | date | date | date |
| Deletes files | del | rm | rm *thisfile.txt* |
| "Echoes" output to the screen | echo | echo | echo *this message* |
| Edits text files | edit | gedit([[a]](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html" \l "FTN.AEN8296)) | gedit *thisfile.txt* |
| Compares the contents of files | fc | diff | diff *file1* *file2* |
| Finds a string of text in a file | find | grep | grep *word or phrase* *thisfile.txt* |
| Formats a diskette | format a: (if diskette is in A:) | mke2fs | /sbin/mke2fs /dev/fd0 (/dev/fd0 is the Linux equivalent of A:) |
| Displays command help | *command* /? | man or info | man *command* |
| Creates a directory | mkdir | mkdir | mkdir *directory* |
| Views contents of a file | more | less([[b]](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html" \l "FTN.AEN8362)) | less *thisfile.txt* |
| Renames a file | ren | mv([[c]](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html" \l "FTN.AEN8374)) | mv *thisfile.txt* *thatfile.txt* |
| Displays your location in the file system | chdir | pwd | pwd |
| Changes directories with a specified path (*absolute path*) | cd *pathname* | cd *pathname* | cd */directory/directory* |
| Changes directories with a *relative path* | cd.. | cd .. | cd .. |
| Displays the time | time | date | date |
| Shows amount of RAM in use | mem | free | free |
| Notes: a. **Gedit** is a graphical text editor; other editors you can use in place of **Gedit** include **nano** and vi. b. The more pager can also be used to page through a file one screen at a time. c. The mv command can both move a file and, if you want to rename a file in the same directory, "move" that file to the same directory with a new name. | | | |

***[Want to try out Red Hat Enterprise Linux?***[***Download***](https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux/try-it?intcmp=701f20000012ngPAAQ)***it now for free.]***

**Topics:**   [Linux](https://www.redhat.com/sysadmin/topics/linux)