How to use the Window command line (DOS)

This document covers the basic in navigating and using the Microsoft Windows command line. On this page, you'll learn how to move around in the command line, find files, manipulate files, and other important commands. Keep in mind that there are over 100 different commands that have been used in MS-DOS and the Windows command line. If you are interested in learning every command, see our [Microsoft DOS and command prompt help page](http://www.computerhope.com/msdos.htm), which gives a description and example for every command.

**Get into the Windows command line**

Open a Windows command line window by following the steps below. If you need additional information or alternative methods for all versions of Windows, see our [how to get into DOS and Windows command line](http://www.computerhope.com/issues/chdos.htm) page.

1. Click [Start](http://www.computerhope.com/jargon/s/start.htm)
2. In the Search or Run line type **cmd** and press enter.

**Understanding the prompt**

After following the above steps, the Windows command line should be shown (similar to the example below). Typically Windows starts you at your user [directory](http://www.computerhope.com/jargon/d/director.htm). In the example below, the user is Mrhope, so our prompt is C:\Users\Windows>. This prompt tells us we are in the C: [drive](http://www.computerhope.com/jargon/d/drive.htm) (the default drive letter of the [hard drive](http://www.computerhope.com/jargon/h/harddriv.htm)) and currently in the Mrhope directory, which is a [subdirectory](http://www.computerhope.com/jargon/s/subdirec.htm) of the Users directory.



**Key tips**

* MS-DOS and the Windows command line are not [case sensitive](http://www.computerhope.com/jargon/c/casesens.htm).
* The files and directories shown in Windows are also found in the command line.
* When working with a file or directory with a space, surround it in quotes. For example, My Documents would be "My Documents."
* File names can have a [long file name](http://www.computerhope.com/jargon/l/longfile.htm) of 255 characters and a 3 character [file extension](http://www.computerhope.com/jargon/f/fileext.htm).
* When a file or directory is deleted in the command line, it is not moved into the [Recycle Bin](http://www.computerhope.com/jargon/r/recycbin.htm).
* If you need help with any of command type /? after the command. For example, dir /? would give the options available for the dir command.

**Listing the files**

Let's learn your first command. Type **dir** at the prompt to list files in the current directory. You should get an output similar to the example image below. Without using any dir options this is how dir output appears. As can be seen, you are given lots of useful information including the creation date and time, directories (<DIR>), and the name of the directory or file. In the example below, there are 0 files listed and 14 directories as indicated by the status at the bottom of the output.

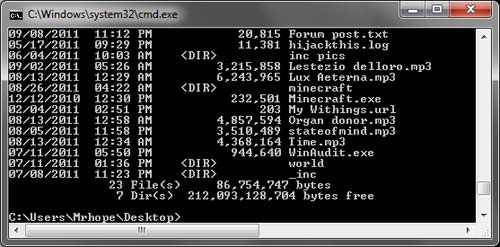


Every command in the command line has options, which are additional switches and commands that can be added after the command. For example, with the dir command you can type **dir /p** to list the files and directories in the current directory one page at a time. This switch is useful to see all the files and directories in a directory that has dozens or hundreds of files. Each of the command options and switches is listed on our [DOS command page](http://www.computerhope.com/msdos.htm). For example, if you want to see all the options for dir see our [dir command page](http://www.computerhope.com/dirhlp.htm) for a full listing.

The dir command can also be used to search for specific files and directories by using wildcards. For example, if you only wanted to list files or directories that begin with the letter "A" you could type **dir a\*** to list only the AppData directory, in this above example. See the [wildcard](http://www.computerhope.com/jargon/w/wildcard.htm) definition for other examples and help with using wildcards.

**Moving into a directory**

Now that we've seen a list of directories (shown below) in the current directory move into one of those directories. To move into a directory, we use the [cd command](http://www.computerhope.com/cdhlp.htm), so to move into the Desktop type **cd desktop** and press enter. Once you've moved into a new directory the prompt should change, so in our example, the prompt is now C:\Users\Mrhope\Desktop>. Now in this desktop directory, see what files are found in this directory by typing the dir command again.



**Understand the files**

Now in the Desktop directory in this example (as shown above), we now have 23 files and 7 directories. As can be seen in the above example, there are many different file types. In Windows, you are familiar with files having icons that help represent the file type. In the command line, the same thing is accomplished by the file extensions. For example, "forum posts.txt" is a [text file](http://www.computerhope.com/jargon/t/textfile.htm) because it has a .txt file extension, Time.mp3 is an [MP3 music file](http://www.computerhope.com/jargon/m/mp3.htm), and minecraft.exe is an [executable file](http://www.computerhope.com/jargon/e/execfile.htm).

* [Listing of file extensions and additional help with file extensions.](http://www.computerhope.com/dosext.htm)

For most users, you'll only be concerned with executable files, which as mentioned above is a file that ends with .exe and are also files that end with .com and .bat. When the name of these files are typed into the command line, the program runs, which is the same as double-clicking a file in Windows. For example, if we wanted to run minecraft.exe typing "minecraft" at the prompt runs that program.

**Note:** Keep in mind that if the executable file you are trying to run is not in the current directory you'll get an error. Unless you have set a [path](http://www.computerhope.com/jargon/p/path.htm) for the directory that contains the executable file, which is how the command line finds [external commands](http://www.computerhope.com/jargon/e/extecomm.htm).

If you want to view the contents of a file, most versions of the command line use the [edit command](http://www.computerhope.com/edithlp.htm). For example, if we wanted to look at the log file hijackthis.log we would type **edit hijackthis.log** at the prompt. For 64-bit versions of Windows that do not support this command you can use the [start command](http://www.computerhope.com/starthlp.htm), for example, type **start notepad hijackthis.log** to open the file in [Notepad](http://www.computerhope.com/jargon/n/notepad.htm). Further information about opening and editing a file from the command line can also be found on the link below.

* [How to open and view the contents of a file on a computer.](http://www.computerhope.com/issues/ch000429.htm)

**Moving back a directory**

You learned earlier the cd command can move into a directory. This command also allows you to go back a directory by typing **cd..** at the prompt. When this command is typed you'll be moved out of the Desktop directory and back into the user directory. If you wanted to move back to the [root directory](http://www.computerhope.com/jargon/r/root.htm) typing **cd\** takes you to the C:\> prompt. If you know the name of the directory you want to move into, you can also type cd\ and the directory name. For example, to move into C:\Windows> type **cd\windows** at the prompt.

**Creating a directory**

Now with your basic understanding of navigating the command line let's start creating new directories. To create a directory in the current directory use the [mkdir command](http://www.computerhope.com/mdhlp.htm). For example, create a directory called "test" by typing **mkdir test** at the prompt. If created successfully you should be returned to the prompt with no error message. After the directory has been created, move into that directory with the cd command.

**Switching drives**

In some circumstances, you may want to copy or list files on another drive. To switch [drives](http://www.computerhope.com/jargon/d/drive.htm) in the Windows command line, type the letter of the drive followed by a colon. For example, if your CD-ROM drive was the D drive you would type **d:** and press enter. If the drive exists the prompt will change to that drive letter.

* [How do you copy files from one drive to another drive?](http://www.computerhope.com/issues/ch001592.htm)
* [Additional information and examples of drive letters.](http://www.computerhope.com/issues/ch000515.htm)

**Creating a new file**

You can create a new file from the command line using the edit command, copy con command, or using the start command to open a file.

* [Complete steps on how to create a file in MS-DOS.](http://www.computerhope.com/issues/ch000398.htm)

**Creating a new batch file**

In the new test directory let's create your first file. In most circumstances, you never need to create any file at the command line, but it is still good to understand how files are created. In this example, we are creating a [batch file](http://www.computerhope.com/jargon/b/batchfil.htm). A batch file is a file that ends with .bat and is a file that can help automate frequently used commands in the command line. We are calling this batch file "example", so type **edit example.bat** at the prompt. As mentioned in the document on creating a file, if the edit command does not work with your version of Windows, use the [start command](http://www.computerhope.com/starthlp.htm) to open the batch file in Notepad. To perform this action, you type **start notepad example.bat** into the prompt.

Both of the above commands open a new blank example.bat window. In the file, type the below three lines, which clear the screen with the [cls command](http://www.computerhope.com/clshlp.htm) and then run the dir command.

@echo off  
cls  
dir

After these three lines have been typed into the file save and exit the file. If you are in the edit command click File (or press Alt+F) and then Save. After the file has been saved and you are back into the command prompt, typing dir should display the example.bat in the test directory.

Now run the batch file to get a better understanding of what a batch file does. To run the batch file type **example** at the prompt, which executes the batch file and clears the screen and then runs the dir command to display the directory listing of the test directory.

* [Full information and additional examples on batch files.](http://www.computerhope.com/batch.htm)

**Moving and copying a file**

Now that we've created a file let's move it into an alternate directory. To help make things easier, create another directory for the files. So, type **mkdir dir2** to create a new directory in the test directory called dir2. After the new directory has been created, use the [move command](http://www.computerhope.com/movehlp.htm) to [move](http://www.computerhope.com/jargon/m/move.htm) the example.bat file into that directory. To do this type **move example.bat dir2** at the prompt, if done successfully you should get a message indicated the file was moved. You could also substitute the move command for the [copy command](http://www.computerhope.com/copyhlp.htm) to [copy](http://www.computerhope.com/jargon/c/copy.htm) the file instead of moving it.

**Rename a file**

After the file has been moved into the dir2 directory, move into that directory with the cd command to rename the file. In the dir2 directory use the [rename command](http://www.computerhope.com/renamehl.htm) to rename the example file into an alternate name. Type **rename example.bat first.bat** at the prompt to rename the file to first.bat. Now when using the dir command you should see the first.bat as the only file.

**Tip:** When renaming any file make sure the file has the same file extension. If you were to rename the .bat file to a .txt file, it is no longer an executable file only a text file. Also, keep in mind that renaming the file to a different file extension does not convert the file. For example, if you were to name the file to a .MP3 file it may look like an MP3 audio file in Windows, but it is not going to play music.

**Deleting a file**

Now that we've had our fun with our new file, delete the file with the [del command](http://www.computerhope.com/delhlp.htm). Type **del first.bat** to delete the first.bat file. If successful, you are returned to the prompt with no errors and the dir command shows no files in the current directory.

**Tip:** When deleting files you can also use [wildcards](http://www.computerhope.com/jargon/w/wildcard.htm) to delete multiple files at once. For example, if the directory contained several .GIF image files you could type del \*.gif to delete all files ending with the .gif file extension.

**Renaming a directory**

Go back one directory to get back into the test directory by using the **cd..** command mentioned earlier. Now rename our dir2 directory to something else using the same rename command we used earlier. At the prompt, type **rename dir2 hope** to rename the directory to hope. After this command has been completed, type dir and you should now see one directory called hope.

**Removing a directory**

While still in the test directory, remove the hope directory by using the [rmdir command](http://www.computerhope.com/rmdirhlp.htm). At the prompt, type **rmdir hope** to remove the hope directory.

**Tip:** If the directory you are trying to remove contains any files or directories, you'll receive an error. To prevent this error use the /s option. For example, if the hope directory still had the first.bat file you would need to type **rmdir /s hope** at the prompt.

**Running a program**

Any file that is an [executable file](http://www.computerhope.com/jargon/e/execfile.htm) can be run from the command line by typing the name of the file. For example, if you listed files using the dir command and see a file named "myfile.exe" typing "myfile" at the command line runs that program.

**How to list available commands**

After getting a good understanding of using the command line from the steps shown above you can move on to other available commands by typing **help** at the command line. Typing "help" gives you a listing of available commands with a brief description of each of the commands.

**Closing or exiting the command line window**

After you are done with the Windows command line, you can type **exit** to close the window.

**In conclusion**

You should now have a good understanding how to navigate the command line, create directories and files, rename directories and files, and delete. As mentioned earlier, there are hundreds of other commands that can be used at the command line. If you want to expand your knowledge even more, we recommend looking at the options available for each of the above commands and go through our [commands overview](http://www.computerhope.com/overview.htm). You can also use our search to find any command by the name of the command or by the action it performs.

Microsoft DOS and Windows Command Line

Short for **Microsoft Disk Operating System**, **MS-DOS** is a non-graphical [command line](http://www.computerhope.com/jargon/c/commandi.htm) [operating system](http://www.computerhope.com/jargon/o/os.htm) created for IBM compatible computers. MS-DOS was first introduced by [Microsoft](http://www.computerhope.com/comp/msoft.htm) in August [1981](http://www.computerhope.com/history/1981.htm) and was last updated in [1994](http://www.computerhope.com/history/1994.htm) with MS-DOS 6.22. Although the MS-DOS operating system is rarely used today, the command [shell](http://www.computerhope.com/jargon/s/shell.htm) commonly known as the **Windows command line** is still widely used. [Continue reading >>](http://www.computerhope.com/jargon/m/msdos.htm)

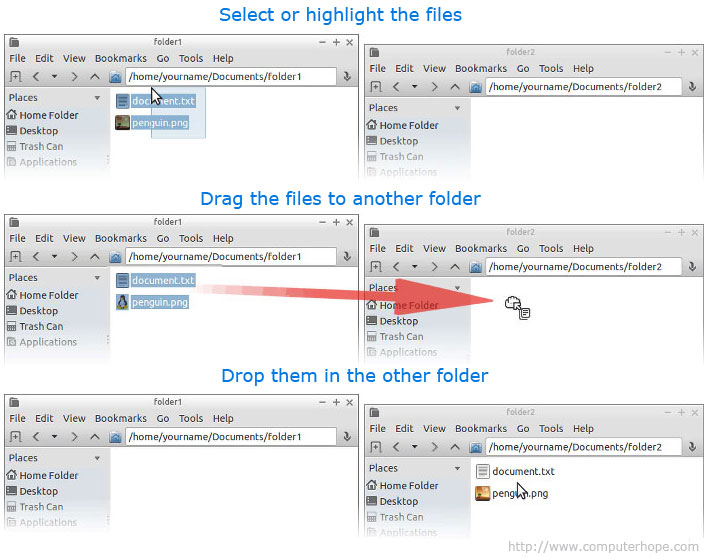
**Tip:** If you are new to MS-DOS or the Windows command line, we suggest starting with [how to use the Windows command line (DOS).](http://www.computerhope.com/issues/chusedos.htm)

**About the command-line shell**

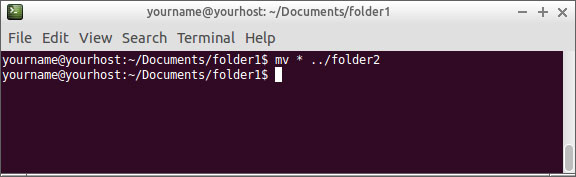
The **command-line shell** is a text-based [user interface](http://www.computerhope.com/jargon/u/ui.htm) for your [operating system](http://www.computerhope.com/os.htm). Unlike a [GUI](http://www.computerhope.com/jargon/g/gui.htm) shell, which uses a graphical representation of the system's services and resources, the command-line shell uses only text. It is the fundamental interface between you and the operating system, and in many cases offers you more direct control over the system processes.

**What Is The Shell Used For?**

When it comes to communicating with your operating system, anything you can do with a GUI can be done at the command line. For example, let's take something you need to do all the time; move files around within your [file system](http://www.computerhope.com/jargon/f/filesyst.htm). Using a GUI, you could select the [icons](http://www.computerhope.com/jargon/i/icon.htm) representing your files, [drag](http://www.computerhope.com/jargon/d/drag.htm) them to another [folder](http://www.computerhope.com/jargon/f/folder.htm), and [drop](http://www.computerhope.com/jargon/d/dragdrop.htm) them there to move them.



However, using the command line, this operation can be performed using a single command:



The command line, once you get used to it, allows you to control your system with greater efficiency and precision than with a GUI.

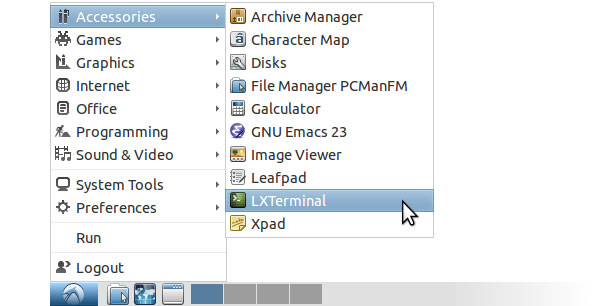
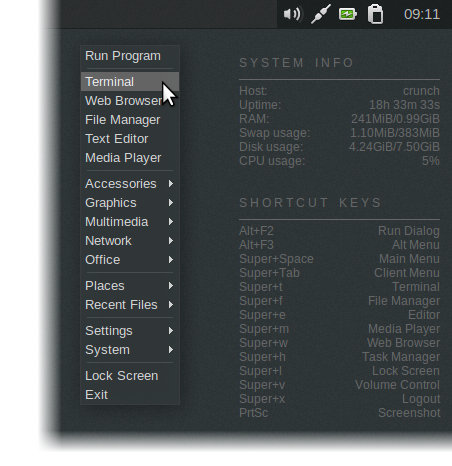
**Why Is It Called A "Shell"?**

The user interface is referred to as a "shell" because it is the outer layer separating you from the innermost parts of the operating system, called the [kernel](http://www.computerhope.com/jargon/k/kernel.htm). Technically, your operating system's GUI is also a "shell," but the command line is frequently referred to casually as "the shell."

**How Do I Get Into The Command-Line Shell?**

The most common way to access the command-line shell is using a terminal emulator.

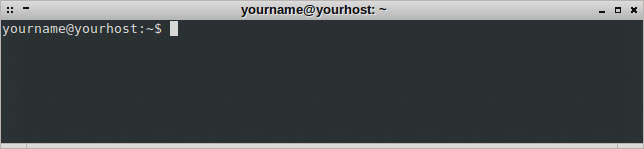
If you are using a modern Linux distribution, chances are it boots into [X Window System](http://www.computerhope.com/jargon/x/xwin.htm)(or "X"), which allows you to run programs in their own windows and interact with them using the mouse. Within X, There are different ways to launch a terminal emulator, but here are the most common:

* Many Linux distributions use a Desktop environment similar to [Microsoft Windows](http://www.computerhope.com/jargon/w/windows.htm), which has a "Start Menu." From this menu, your terminal application is usually located in under "Accessories." For example, [Lubuntu](http://www.computerhope.com/jargon/l/lubuntu.htm) comes with the LXTerminal terminal emulator already installed, and the [shortcut](http://www.computerhope.com/jargon/s/shortcut.htm) is located in the Accessories menu:  
    
    
    
  Selecting this menu item launches the terminal emulator and place you at a command line.
* Different versions of Linux use different kinds of window managers, each with a slightly different interface. With many window managers, [right-clicking](http://www.computerhope.com/jargon/r/righclic.htm) the [Desktop](http://www.computerhope.com/jargon/d/desktop.htm)shows a menu that contains a terminal option. For instance, the right-click menu in Crunchbang Linux has a "Terminal" option. Here's what that menu looks like, after right-clicking anywhere on the Desktop:  
    
    
    
  Here, the Accessories menu would also have an option for running the terminal.
* On many versions of Linux, pressing the **Super Key** and "**T**" at the same time is the default hotkey for launching a terminal. (The Super Key is usually the first key to the left of the spacebar, also known as the [Windows key](http://www.computerhope.com/jargon/w/winkey.htm)).

If you cannot figure out how to launch your terminal emulator, see the documentation for your own particular operating system, and search for "terminal".

**The Command Prompt**

When you launch the terminal emulator, you are presented with a command [prompt](http://www.computerhope.com/jargon/p/prompt.htm).



**Note:** These examples are all performed using the Bourne-Again Shell ([**bash**](http://www.computerhope.com/unix/ubash.htm)), which is the default command-line shell on [Linux](http://www.computerhope.com/jargon/l/linux.htm).

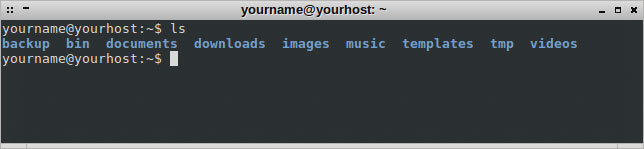
Let's look at this command prompt closely. The prompt can be configured to be anything you like, but by default, it looks like what we see here, providing us with useful information before we type in a command.

Specifically, it tells us *who we are* ([username](http://www.computerhope.com/jargon/u/username.htm)) and *where we are* (what system we are using and our working directory). Here are what the individual parts of the prompt mean:

|  |  |
| --- | --- |
| yourname | Your username. |
| @ | An [at symbol](http://www.computerhope.com/jargon/a/at.htm) separates the username and [hostname](http://www.computerhope.com/jargon/h/hostname.htm). |
| yourhost | The system hostname. |
| : | The [colon](http://www.computerhope.com/jargon/c/colon.htm) is the character that separates the hostname and the [working directory](http://www.computerhope.com/jargon/c/currentd.htm). |
| ~ | Our current ("working") directory. The [tilde](http://www.computerhope.com/jargon/t/tilde.htm) ("**~**") is a special character which represents our [home directory](http://www.computerhope.com/jargon/h/homedir.htm). When you start a shell session, this is the directory where you typically begin working by default. |
| $ | A special character which indicates that we do not have [superuser](http://www.computerhope.com/jargon/r/root.htm) privileges. If logged in as **root** (the superuser), this character changes to a **#**. |

**Entering Commands**

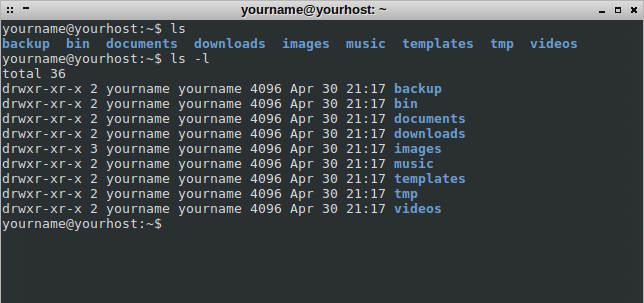
Let's type in our first command. First things first: let's see what files are in our home directory. The most basic way to do this is with the [**ls**](http://www.computerhope.com/unix/uls.htm) command. Type **ls** at the command prompt and press enter.



As shown above, the ls command lists the contents of the directory and places us at another command prompt.

**Note:** All commands and [file names](http://www.computerhope.com/jargon/f/filename.htm) in Unix and Linux are [case-sensitive](http://www.computerhope.com/jargon/c/casesens.htm). The **ls** command, for example, *must* be typed in lowercase letters, just as it is above.

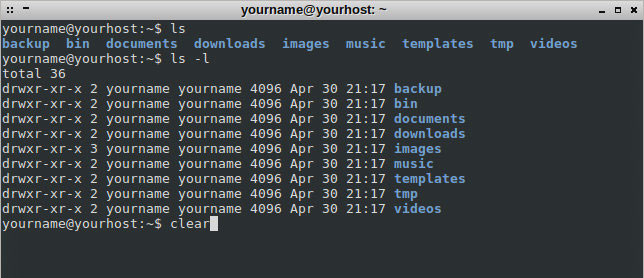
For a more detailed listing of the directory contents, we can specify *command-line options*, also known as *switches*. These are additional arguments to the command that modify the way it operates. Here, we will use the **-l** (use long list format) option to get more information from our listing:



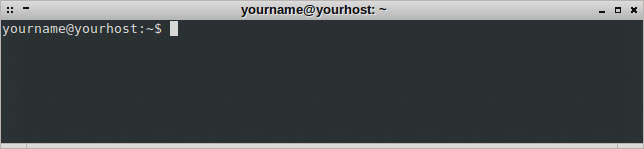
As shown above, the ls -l command provides us with [a lot more information](http://www.computerhope.com/unix/uchmod.htm#05) about the contents of our home directory. At the moment, the most important information we should notice is that the first letter of the listings is a **d**, meaning that each of those directory contents is itself a directory.

**Clearing the screen**

Before we do anything else, let's clear the screen, to remove the clutter produced by the output from our previous commands. Use the [**clear**](http://www.computerhope.com/unix/uclear.htm) command by typing **clear** at a new command prompt and press **enter**.



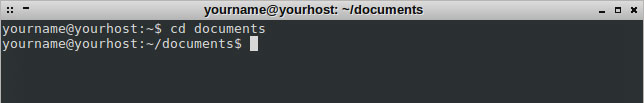
After pressing enter, the screen clears and places you at a new command prompt at the top of your terminal screen.



**Changing Directory**

Let's move into our **documents** folder using the [**cd**](http://www.computerhope.com/unix/ucd.htm) ("change directory") command. To change the current directory, type "**cd**" and then a space, then the name of the directory you would like to change to and then press enter.

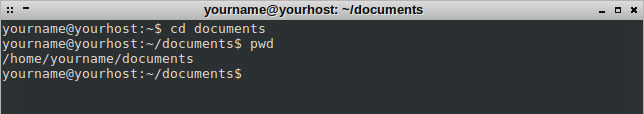
**cd** assumes that the name of any directory you specify is relative to your current directory. Therefore, to change into the **documents** directory which exists in our current directory, type "**cd documents**":



The command prompt now reflects the new working directory: **~/documents**. This is equivalent to the full pathname: **/home/yourname/documents**. No matter what user you are logged in as, "**~**" always represent your home directory. In the example, the new working directory is also seen in the [title bar](http://www.computerhope.com/jargon/t/titlebar.htm) of our terminal window.

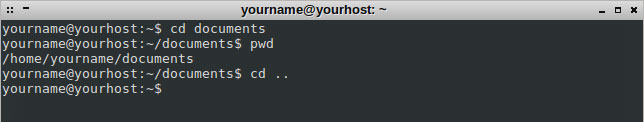
**Note:** Unlike [MS-DOS](http://www.computerhope.com/jargon/m/msdos.htm) and Microsoft Windows, [hierarchical](http://www.computerhope.com/jargon/h/hierfile.htm) directory names in Linux are separated by a [forward slash](http://www.computerhope.com/jargon/f/forwards.htm) ("**/**") rather than a [backslash](http://www.computerhope.com/jargon/b/backslash.htm) ("**\**"). This change of symbols is a fundamental difference between these various operating systems.

At any time, you can verify your current directory by using the [**pwd**](http://www.computerhope.com/jargon/p/pwd.htm) ("print working directory") command:



The command prints the name of the current directory to the terminal, in this case "**/home/yourname/documents**". Again, with "**~/documents**" the "**~**" represents our home directory, which is **/home/yourname**. So if we are in our home directory, typing "**cd documents**", "**cd /home/yourname/documents**", or "**cd ~/documents**" would all place us in the same directory.

Now let's move up one directory, back into our home directory. The special directory name "**..**" (the double-dot) means "one directory up":



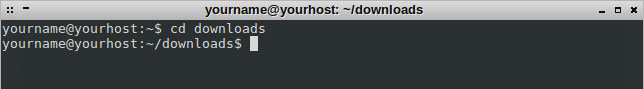
we are now back in our home directory.

**Note:** If you are more familiar with MS-DOS, keep in mind that there must be a space between cd and the two periods.

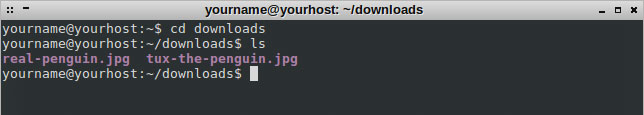
**Moving Files**

Now let's get familiar with how to move files around.

Let's say we just downloaded two images with a [web browser](http://www.computerhope.com/jargon/b/browser.htm), and they saved into our **downloads** folder. Let's go in there and take a look. First, we change directory into the **downloads** directory.



Then list the files in the directory with the [**ls**](http://www.computerhope.com/unix/uls.htm) command.



Here we see two [JPEG](http://www.computerhope.com/jargon/j/jpeg.htm) files. Let's get them out of the **downloads** folder, and put them someplace more appropriate. Let's put them in our **images** folder instead. To do this, we'll use the [**mv**](http://www.computerhope.com/unix/umv.htm) ("move") command.

The **mv** command takes the following general form:

mv [*OPTIONS*...] *SOURCE* *DESTINATION*

(*OPTIONS* is enclosed in brackets because you are not required to specify any options to complete the command. The *SOURCE* and *DESTINATION* arguments are both required).

*SOURCE* is the name of the file or files that you want to move. *DESTINATION* is where you want to move them. If *DESTINATION* is a directory name, the file or files are moved into that directory.

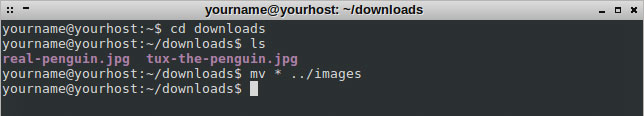
we are currently in the directory **/home/yourname/downloads** and we want to move the files into **/home/yourname/images**. We could specify our *DESTINATION* using the full pathname ("**/home/yourname/images**"), or as a pathname relative to our current directory ("**../images**"), or using a tilde as an abbreviation for our home directory ("**~/images**"). All of these are equivalent and give the same result. Let's use the relative pathname for our *DESTINATION*: **../images**.

So, how will we specify our *SOURCE*?

We want to move both of the files, and they are the only files in the directory. So, the simplest way to specify them would be to tell **mv** "move all of the files in this directory into **../images**." We can do this using a [wildcard](http://www.computerhope.com/jargon/w/wildcard.htm).

A wildcard is a special character which can represent more than one file name (or more than one *part* of the file name). Here, we are going to use the [asterisk](http://www.computerhope.com/jargon/a/asterisk.htm) wildcard ("**\***"). If we use **\*** as our entire file name, the shell applies our command to *every file* in our working directory.

So, our command will read as follows: "**mv \* ../images**". Let's enter it now.



It looks like nothing has happened, but that is the shell's way of telling us no errors occurred. If there was a problem, it would display an error message or warning of some kind.

Let's verify that the files have moved. We can list the contents of another directory by running [**ls**](http://www.computerhope.com/jargon/a/asterisk.htm) and specifying a pathname. In this case, our pathname is "**../images**":

Lising files in parent directory in a shell

The images are both there, along with another directory that already existed ("**wallpapers**"). We should make a new directory of our own, to better organize our images and keep the main **images** directory from getting cluttered. Let's do that now.

**Making Directories**

Let's change into the **images** directory where our two penguin images now reside.

Moving into the images directory in a Linux shell

Now let's make a new directory for our penguin images. Creating new directories is accomplished with the [**mkdir**](http://www.computerhope.com/unix/umkdir.htm) command.

The simplest form of the **mkdir** command is to give it one argument: the name of the new directory. We'd like to make a new directory within our current directory, so we do not have to give **mkdir** a full pathname, just the name of the directory itself. Let's call our new directory **penguin-pix**.

So, the command we are going to enter is "**mkdir penguin-pix**" to make a new directory.

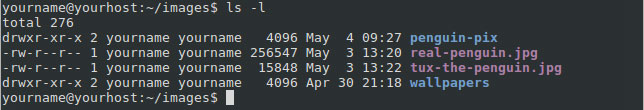
Using the mkdir command to make a directory in a Linux shell

Now let's verify it is there, by listing the directory contents:

Listing the files  in the images direcotry using ls command

**Note:** You can see that when we list the directory contents, file names and directory names are different colors to help identify the different files and directories. Although color highlighting is common with most shells, not all shells do this by default.

If we could not tell by the color of the names we could always check which of these files are directories by invoking **ls** with the **-l** (long list) option.



The additional information includes [permissions](http://www.computerhope.com/jargon/p/permissi.htm) (who has access to the file and what kind of access), [ownership](http://www.computerhope.com/jargon/o/owner.htm) (who is the file's owner), authorship (who created the file), file size, and mtime (date the file was modified last). The first letter on the lines for **penguin-pix** and **wallpapers** is a "**d**", indicating that those are directories. For more information about file listings, see the documentation of the [**chmod**](http://www.computerhope.com/unix/uchmod.htm#05) command.

We can now move our images into the new directory. To tell **mv** what files to move, we can use a wildcard again, but this time let's make sure it knows to move only the image files, not the directories. Since both image files end with the JPEG extension **.jpg**, we can specify the file name "**\*.jpg**": this tells the shell "the files I want to move are any files in the current directory that end with **.jpg**". So our command will look like this: "**mv \*.jpg penguin-pix**":

Moving all JPEG files in the current directory to another directory in a Linux shell

No error message was given, meaning the command completed successfully. Let's verify the files were moved by listing the contents of **penguin-pix**:

Listing all the files of a directory

**Removing Files**

We have two penguin pictures, but perhaps we decided that we only need **real-penguin.jpg**, and we do not need **tux-the-penguin.jpg** anymore. We can delete this file by using the [**rm**](http://www.computerhope.com/unix/urm.htm) ("remove") command.

We'll need to provide **rm** with the file name to delete. We are currently in the **images** directory, but our image is now in the **penguin-pix** subdirectory. We could change our working directory to **penguin-pix** and then run the command, but that is only extra work, we can skip that step by using the directory as part of the path.

Our command will look like this: "**rm penguin-pix/tux-the-penguin.jpg**":

Removing a file in a Linux shell

Once again, the shell executes our command without giving any feedback; the command executed successfully, so the shell returns us to the command prompt, letting us know that it is ready for another command.

**Warning:** Unlike dragging a file to the [trash](http://www.computerhope.com/jargon/t/trash.htm) or to the Microsoft Windows [recycle bin](http://www.computerhope.com/jargon/r/recycbin.htm), there is no way to [undo](http://www.computerhope.com/jargon/u/undo.htm) a file deletion with **rm**. Remove your files with care! Once gone, it is (virtually) impossible to get them back.

Let's just verify that the file is really gone:

Verifying a file has been deleted

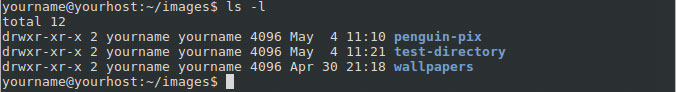
**Removing Directories**

That is great for files, but what about [directories](http://www.computerhope.com/jargon/d/director.htm)? By default, **rm** only removes files and not directories.

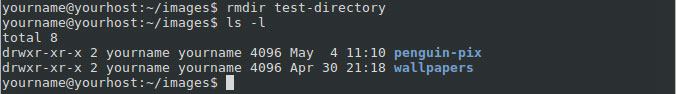
The [**rmdir**](http://www.computerhope.com/unix/urmdir.htm) ("remove directory") command removes a directory, assuming that it is empty. Let's test it by creating a new, empty directory, using the "**mkdir test-directory**" command.

Make a new directory

Let's verify it is there with "**ls -l**" command.



We can now remove our brand new directory using the command "**rmdir test-directory**", and check that it is gone by running the "**ls -l**" command again.



Yup, it is gone.

However, what would happen if we tried to "**rmdir penguin-pix**"?

Getting an error that rmdir failed to remove a directory

We cannot **rmdir** that directory because it is not empty.

If you are sure you want to remove **penguin-pix** and everything inside it (including all files and any directories it might contain), you can use the **rm** command with the **-r** option. Normally, **rm** will not operate on a directory:

Getting an error that rmdir failed to remove a directory

However, if invoked with the **-r** option, **rm** removes a directory [recursively](http://www.computerhope.com/jargon/r/recursive.htm), which means it travels into the directory and any [subdirectory](http://www.computerhope.com/jargon/s/subdirec.htm), deleting everything including the directory. So, if we run the command "**rm -r penguin-pix**" it removes the directory, and the file that was inside. If there were any directories inside, they would also have been deleted.

Using the -r command to recursively remove a directory

**Warning!** Once again, there is no undo for this command. When you **rm -r** a directory, that directory and everything inside of it is removed forever! Like most Linux commands, **rm** is very powerful and should be used with care.

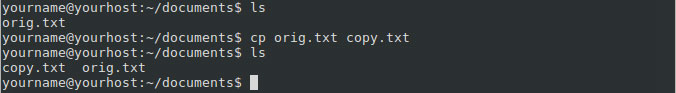
**Copying Files**

To [copy](http://www.computerhope.com/jargon/c/copy.htm) a file, use the [**cp**](http://www.computerhope.com/unix/ucp.htm) command. The simplest form of **cp** is:

cp *SOURCE* *DESTINATION*

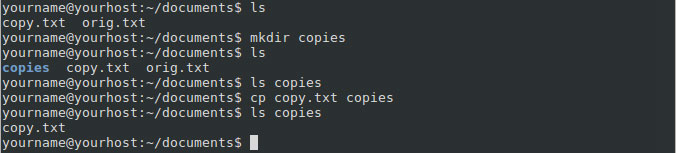
*SOURCE* is the name of the file you want to copy, and *DESTINATION* is the pathname specifying the destination of the new file. The destination can be a new file name or the name of the directory where you would like a copy to be located.

Let's say we have a [text file](http://www.computerhope.com/jargon/t/textfile.htm) in our **documents** folder called **orig.txt**, and we want to make a copy of it called **copy.txt**. The command we should use is "**cp orig.txt copy.txt**":



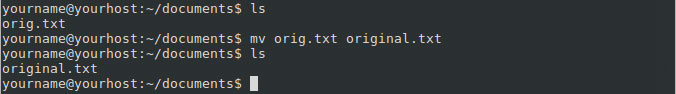
If, on the other hand, we specify a directory as the destination of the **cp** command, it makes a copy with the same name as the original in the destination directory. In the next example, we do six things:

1. list the contents of the working directory;
2. make a new directory, called **copies**;
3. list the contents of the working directory, which now includes the **copies** directory;
4. list the contents of the **copies** directory (there are none. It is empty);
5. copy **copy.txt** into the **copies** directory;
6. list the contents of the **copies** directory, which now contains a copy of **copy.txt**.



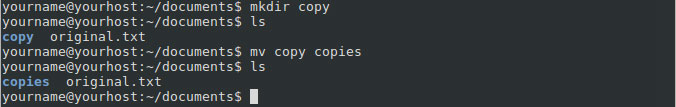
**Renaming A File**

The [**mv**](http://www.computerhope.com/unix/umv.htm) command can also be used to change the name of a file. For example, the command "**mv orig.txt original.txt**" will rename the file **orig.txt** to **original.txt**:



**Renaming A Directory**

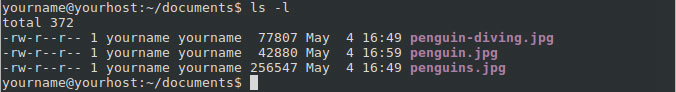
Renaming directories works exactly like renaming files, with the **mv** command. Here, we create a directory named "**copy**", but rename it to "**copies**" using the command "**mv copy copies**":



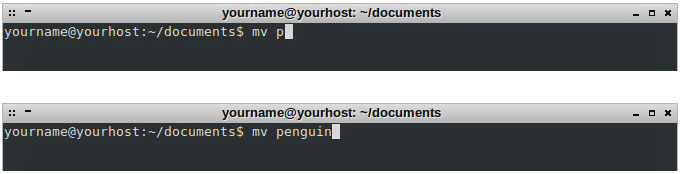
If the **copy** directory contained any files or directories, they would be unchanged. The only thing that would change is the name of the directory that contained them.

**Auto-Completion**

Most Linux command-line shells automatically complete the word you have started typing if you press the [TAB](http://www.computerhope.com/jargon/t/tab.htm) key. Auto-completion can help save you time, especially with long file names. For example, let's say you have three files in your documents folder: **penguin-diving.jpg**, **penguin.jpg**, and **penguins.jpg**:

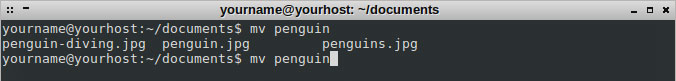


If you type a command and then type "**p**" and press the **TAB** key the shell completes the word "penguin" for you.



It stops there because all the files start with the word **penguin**, but after that it is not sure which file you would like to specify.

If you are in a situation where there is more than one autocomplete match for what you have already typed, pressing **TAB** twice prompts the shell to list all the possible matches. So in the situation above where you typed **p** and **TAB** and the shell autocompletes **penguin** for you. Press **TAB** two more times would give you all possible options, and place you back at the prompt, with your command filled in where you left off:



Autocompletion also works for completing path names and command names.

**Exiting The Shell**

Typing the **exit** command exits the command shell, and either close the terminal window, or prepare the terminal window to be closed manually.