

How to Use the Command Prompt to Navigate – and Perhaps Repair – Your PC

This article shows you how to:

- Find your way to the Command Prompt when needed
- Understand the basics of putting commands together
- Learn the essential commands you'll need to use

It's not something most of us use by choice, but there are certain technical aspects of Windows that can only be accessed using the Command Prompt. More pressingly, if Windows won't start and you have to use a recovery drive, you might find yourself forced to get acquainted with the Command Prompt to stand any chance of fixing the problem.

In this article, I'll explain the basics of the Command Prompt, and I'll introduce you to all of the most useful commands you'll need to know to find your way around it and achieve something useful.



Contents:

A Quick Introduction to the Command Prompt.....	U 435/2
Command Prompt Basics: Commands, Arguments and Switches.....	U 435/3
How to Find the Command Prompt When You Need It	U 435/6
Getting Around: Navigating Your Drives and Folders.....	U 435/8
Essential Commands for Getting Things Done	U 435/14

A Quick Introduction to the Command Prompt

Before Windows: In the early days of the PC, there was no such thing as Windows. Instead, your operating system was MS-DOS (short for Microsoft Disk Operating System). It gave you a plain black screen on which you had to type commands. If you made the slightest typing mistake, you'd see a sniffy error message and have to try again.

MS-DOS

MS-DOS was hard to love, and the arrival of Windows with its friendly point-and-click interface was a welcome relief to most PC users.

The Command Prompt is the modern equivalent

But MS-DOS has never really quite gone away. Even in the latest versions of Windows, it still lurks in the background in the form of the Command Prompt, available if you want it. Most of the time you probably don't, but sometimes it can't be escaped.

It can be useful when trying to solve problems...

There are certain aspects of Windows that are only available 'from the command line' – in other words, by typing commands into a Command Prompt window. Those aspects are generally only needed for very specialised purposes, but it's in the nature of computing that we can all find ourselves in peculiar situations at times, in need of equally-peculiar solutions!

In addition, some modern programs either have to be used from a Command Prompt or provide extra options when used that way. As an example, some of the best utilities for converting audio and video from one format to another have to be used from the command line.

... particularly if Windows won't start

But the Command Prompt really comes into its own when you have nowhere else to turn. If Windows won't start, for instance, you should be able to find your way to various types of help by starting the PC from a recovery drive or a Windows installation DVD. That gives you access to various tools, and one of those is the Command Prompt. That's the only one that gives you access to the

files on your hard drive (and other drives), making it possible to fix whatever has gone wrong – or at least rescue your personal files by copying them to another disk. Quite often, you'll probably be following instructions you've found elsewhere, but if nothing else it's useful to have some understanding of what you're actually doing!

Command Prompt Basics: Commands, Arguments and Switches

You use the Command Prompt by typing commands. There are dozens of commands built into Windows, some more commonly used than others, and they all have one-word names – usually abbreviations. For example, there's **del** (delete), **ren** (rename) and **chkdsk** (check disk). You type a command, press **Enter** and (assuming you typed it correctly!) Windows runs that command.

Using commands

You're probably aware that file and folder names in Windows are not case-sensitive, and the same applies throughout the Command Prompt. Whether you type **chkdsk** or **CHKDSK** or even **chKdSk**, you'll always get the same result. Generally, though, commands are typed in lowercase since there's no point in reaching for the Shift key if nothing is taking any notice of it.



Some of these commands are self-contained, needing nothing more. For instance, if you type the command **help** and press **Enter** you'll see a list of all the available commands with a one-line description of each.

But the majority of commands need more information. Let's take two of those I mentioned above:

- **del**: delete something... but what?

- **ren:** rename something... but again, what? And what new name do you want to give it?

*Many commands
need more
information*

The extra items of information you supply are known as 'arguments' and they're separated from the command (and each other) by spaces. Here, for instance, we're asking to delete a file named `myfile.txt`:

del myfile.txt

And here we're renaming `newfile.txt` to `oldfile.txt`:

ren newfile.txt oldfile.txt



That leads to an essential point to remember about arguments. If an argument contains a space, it must be enclosed in "double quotes". Otherwise Windows will think the space denotes the end of the argument and will ignore (or be confused by) whatever comes next.

To illustrate this, imagine you want to delete a file named 'my file.txt'. If you type this command:

del my file.txt

you'll see an error message saying that no file named 'my' could be found. Instead, you have to type this:

del "my file.txt"

*Switches specify
how the command
should work*

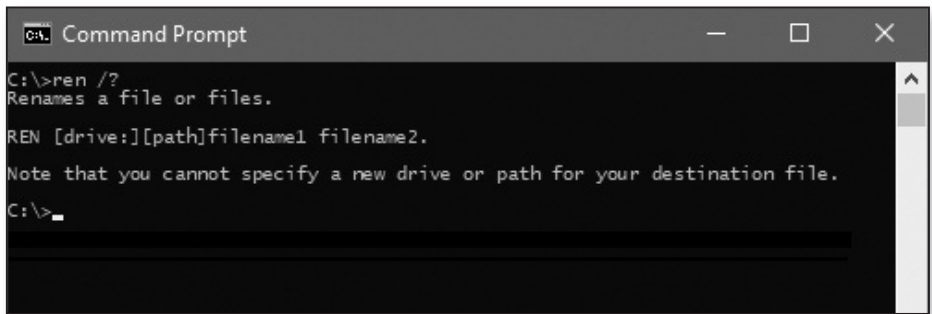
There are also special built-in types of argument called 'switches'. These are generally used to specify exactly how the command should be interpreted. A switch is almost always prefixed with a / sign and often consists of just a single letter.

To give you an example, the **chkdsk** command I mentioned earlier checks a disk for errors and displays a report of what it found at the end. But by adding a particular switch, `/r`, it will also repair any errors it finds along the way:

chkdsk /r

There's one particular switch that's well worth knowing and committing to memory: the question mark. If you type any command and add the `/?` switch, you'll see instructions for how to use that command. In the screenshot below, I've used `ren /?` for help with the `ren` command:

Create groups of related tiles



```
C:\>ren /?
Renames a file or files.

REN [drive:][path]filename1 filename2.

Note that you cannot specify a new drive or path for your destination file.

C:\>
```

That help does take some getting-used-to: things aren't explained in quite the way you might have hoped, but it's really a matter of familiarity. If you know you need to use a particular command, and you don't know where else to turn for help, it's certainly better than nothing.

In a nutshell, the help gives a short explanation of what the command does, then an example of how it's used, then a list of any available switches and their meanings. That might be followed by additional notes about how to use the command or limitations to using it.

In the usage example, anything enclosed in square brackets is optional: you may not want or need to include it. Everything outside square brackets is required.

As you can see in the screenshot above, when renaming a file, all that's absolutely required is that you provide the name of a file to be renamed ('filename1') and the new name you want to give it ('filename2').

How to Find the Command Prompt When You Need It

Open the Command Prompt in Windows

The Command Prompt is one of the many accessories built into Windows and available from the Start menu (or Start screen in Windows 8.1). When you need it, here's where to find it:

- **Windows 10:** open the Start menu, scroll down to the 'W' section and click the **Windows System** folder and you'll find Command Prompt inside it.
- **Windows 8.1:** go to the Start screen and click the circled arrow at the bottom to reach the 'All Apps' screen. Scroll to the right until you reach the **Windows System** heading and Command Prompt is below it.
- **Windows 7:** open the Start menu and click **All Programs**, then open the **Accessories** folder and that's where you'll find Command Prompt.



In any version of Windows, there's an alternative method that works. Press the **Win** key to open the Start menu or Start screen and start typing the word **command**. By the time you reach the third or fourth letter, you should see a link to Command Prompt at the top of the search results.

That all assumes that Windows is running, of course, but what if it isn't? Perhaps the reason you need the Command Prompt is to try to fix a problem that's preventing Windows from starting. In that case, you'll need to do the following:

Windows 10 or 8.1:

Find the Command Prompt if Windows won't start

If Windows is trying and failing to start, let it keep trying. After three failed attempts to start, it should lead you to a screen headed **Automatic Repair**. On this screen, click the **Advanced options** button which takes

you to a screen headed **Choose an option**. From this screen, click on **Troubleshoot**, then **Advanced options**, then **Command Prompt**.

But perhaps you never reach that Automatic Repair screen: however often you try to get Windows started, nothing happens. In that case, you'll have to boot the PC using a recovery drive. You can find all the details of this in article **R125 – Recovery Drive: Repair Your Windows 10 PC**, which was included in update 5/20. This takes you to the **Choose an option** screen mentioned above, where again you choose **Troubleshoot**, then **Advanced options**, then **Command Prompt**.

Windows 7:

If you can't get Windows to start, you need to start your PC using a System Repair Disc. This is something you can create in Windows 7 (provided you do it while Windows is working properly, of course!) by opening the Start menu, typing **system repair** and then clicking the link to **Create a System Repair Disc**.

Use a System Repair Disk in Windows 7

When you start your PC using the System Repair CD or DVD, you'll be prompted to change your language and input method to UK English (if necessary) and System Repair will then search for Windows installations on your PC. When it's finished doing that, click **Next**. On the next screen, choose **Use recovery tools that can help fix problems starting Windows** and click **Next**, then choose **Command Prompt**.

In the details above, I've explained how to get straight to Command Prompt, since that's what this article is about. However, if your problem is that Windows won't start, there are two other tools worth investigating first on the Recovery Drive or System Repair Disc:



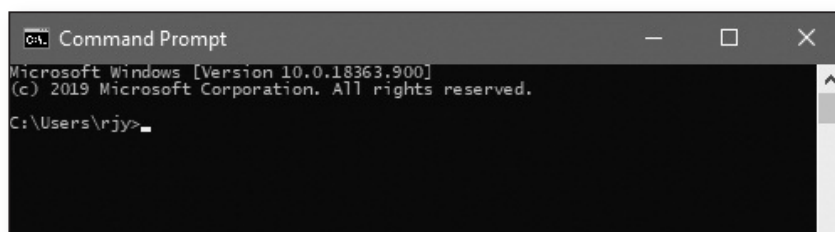
- Startup Repair does what it says on the tin, attempting to fix any problem that's preventing Windows from starting. It won't always work, but it's quick and easy to try.
- System Restore allows you to restore Windows to an earlier point in time by picking your most recent restore point from the list presented.

If neither of those does the trick, I recommend running the **chkdsk** command from the Command Prompt (explained on page 15) to find and fix any errors on your hard disk. I've rescued a good many non-starting PCs this way.

Getting Around: Navigating Your Drives and Folders

The prompt shows you where you are

When the Command Prompt window first appears, it looks like the screenshot below. There's a note of your Windows version and a copyright notice at the top (both of which you can ignore), and then the prompt itself with the text cursor flashing expectantly beside it:



The prompt shows your current path within the Command Prompt – the path to the folder you're currently working with. If you've opened the Command Prompt from Windows, this will be the path to your personal folder, the one containing your 'Documents', 'Pictures', 'Music' (and so on) folders, as in the example above.

If you opened the Command Prompt from a recovery drive, it will show the drive letter of that drive, such as `E:\>`.

To switch to a different drive (because you want to work with folders and files on that drive), just type the drive letter and a colon then press **Enter**. In the example below, I'd arrived at a drive named E: but I want to work with files on my hard drive (C:), so I type `c:` and the prompt then changes to `C:\>` to show that's where I now am:

Switch to a different drive



Having selected the drive you want to work with, a common thing you'll want to do is switch to a particular folder. For this, you use the command **cd** (short for Change Directory).

Switch to a different directory

If you were using computers before the mid-1990s, you'll probably remember the word 'directory'. It's the old-fashioned name for what we now call a 'folder'. There's no difference between the two, but throughout the Command Prompt you'll see references to 'directories' (and the abbreviation 'dir') rather than 'folders'.



Let's assume you're currently looking at a prompt that says `C:\>`, as in the screenshot above, indicating you're at the root of your C: drive. You'd like to go to your personal 'Documents' folder, which in my case is `C:\Users\rjy\Documents`. Here's the slow-but-sure way to get there:

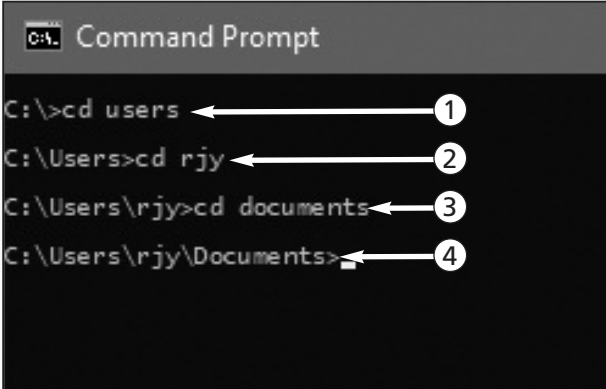
Example: go to your Documents folder

① First type **cd users** and press **Enter**. The system checks whether there's a directory (folder) named 'users' on

your C: drive and (since there is) switches to it, changing the prompt to **C:\Users\>**.

② Now type **cd** and the name of your own personal folder (in my case **rjy**) and press **Enter**. Again, there's a check that this folder exists, and that's where you arrive.

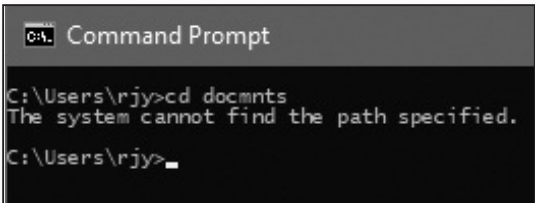
③ Finally, type **cd documents** and press **Enter**. Assuming that exists, you arrive in that folder, with the prompt now indicating that's where you are ④.



```
C:\> Command Prompt
C:\>cd users ← ①
C:\Users>cd rjy ← ②
C:\Users\rjy>cd documents ← ③
C:\Users\rjy\Documents> ← ④
```

*If you make a
typing mistake*

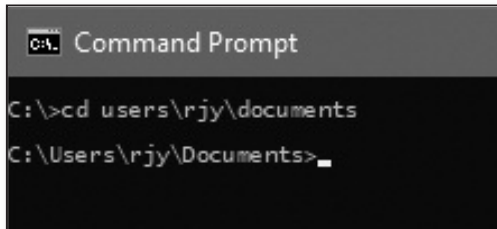
At every step through this routine, you know you've been successful because the prompt changes to show you where you now are. If you mistype the name of a folder (or forget the space after **cd**, or make some other mistake), an error message will appear below the command you just typed and the prompt will remain as it was:



```
C:\> Command Prompt
C:\Users\rjy>cd docmnts
The system cannot find the path specified.
C:\Users\rjy>
```

I mentioned that was the 'slow-but-sure' way of getting around. We moved from folder to folder one step at a time. But we could have done the whole thing at once by specifying the whole path in a single **cd** command:

Navigate more quickly



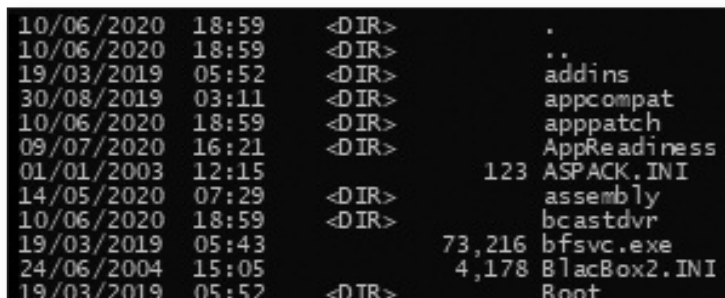
```
C:\>cd users\rjy\documents
C:\Users\rjy\Documents>_
```

There are two reasons why you might not do that. The first is that if you make just one typing mistake in that fairly-long command, you'll have to type the whole thing again after the error message appears. (So that's really a matter of how confident a typist you are!)

The second is that you may not yet know the name of the folder you want to go to next, and that leads to another essential command: **dir**.

Wherever you are, type **dir** and press **Enter** and you'll see a list of all the files and directories (folders) in the current folder. They're presented in alphabetical order, rather than folders first then files, like this example of the partial contents of my Windows folder:

See the contents of a directory



```
10/06/2020 18:59 <DIR> .
10/06/2020 18:59 <DIR> ..
19/03/2019 05:52 <DIR> adds
30/08/2019 03:11 <DIR> appcompat
10/06/2020 18:59 <DIR> apppatch
09/07/2020 16:21 <DIR> AppReadiness
01/01/2003 12:15 123 ASPACK.INI
14/05/2020 07:29 <DIR> assembly
10/06/2020 18:59 <DIR> bcastdvr
19/03/2019 05:43 73,216 bfsvc.exe
24/06/2004 15:05 4,178 BlacBox2.INI
19/03/2019 05:52 <DIR> Boot
```

Each item in the list begins with the date and time it was created, and at the far right you'll see the name of the file or folder. In between, you'll see **<DIR>** if it's a directory, or its size in bytes (such as 73,216) if it's a file.

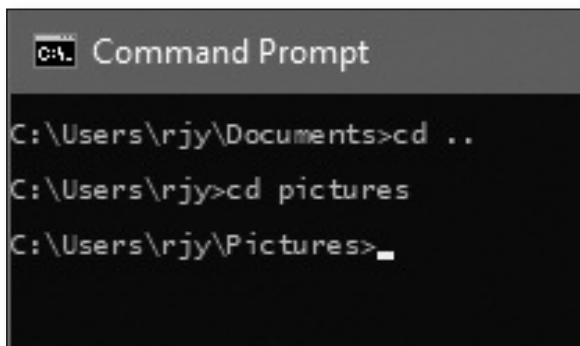
Go back up to the parent directory

Now we're going to return to the **cd** command, as there's one vital element I haven't explained. I gave the example a moment ago of reaching your Documents folder, but what if – having arrived there – you want to go to your Pictures folder. Unlike all the moving around we've done so far, the folder we want isn't inside the folder we're currently working with, so how do we get there?

The trick is to type the **cd** command with two dots, which takes us up to the parent folder:

cd ..

In the example below, I start in my Documents folder. I type **cd ..** which takes me back up to the rjy folder, which is the one containing the Pictures folder, and I can then switch to that:

A screenshot of a Windows Command Prompt window. The title bar is grey and says "Command Prompt". The window has a black background with white text. The text shows the following sequence of commands and directory changes:
C:\Users\rjy\Documents>cd ..
C:\Users\rjy>cd pictures
C:\Users\rjy\Pictures>_

Why navigate around?

So, now you know how to find your way around your drives and folders in the Command Prompt, and that leaves just one question: why would you need to?

Well, sometimes you don't. I'll give you an example that explains why it can be helpful to navigate to a particular folder.

Perhaps I want to delete a file named 'test.txt' from the folder named C:\Windows\System32. Whichever drive or folder the prompt is currently showing, I could do that by typing this command:

```
del c:\windows\system32\test.txt
```

I haven't had to navigate anywhere. But I could have used this pair of commands:

```
cd windows\system32
```

```
del test.txt
```

In the first command I switched to the required folder. In the second, all I had to do was supply the filename I wanted to delete: it's in the current folder, so the system is guaranteed to find it there. Using this method, I've achieved exactly the same, but it's taken me slightly longer, with a little more typing, to do it.

But imagine I wanted to do various things in this System 32 folder: delete files, rename them, create new folders, and so on. If I navigate to that folder first, all future commands can be shorter because I don't have to keep typing the full path every time. Rather than this:

```
del c:\windows\system32\test.txt
```

```
del c:\windows\system32\test2.txt
```

```
ren c:\windows\system32\test3.txt test4.txt
```

I can simply do this:

```
del test.txt
```

```
del test2.txt
```

```
ren test3.txt test4.txt
```

Being in the right directory can save typing

Essential Commands for Getting Things Done

*The commands
you'll find most
useful*

Now you know the basics of using the Command Prompt and navigating around your drives and folders from the command line, it's at your disposal whenever you need it. All you need to know now are the commands you can use with it, so without further ado let's run through all the most useful of those commands in alphabetical order.

ATTRIB – check or change a file's attributes

*The command
attributes of a file*

Files can have a range of 'attributes' (or settings) applied to them, and once in a while you may need to know or change those attributes. There are four common attributes, each represented by a letter (indicated in brackets below):

- **Archive (A):** this is largely meaningless these days; all files have the 'A' attribute.
- **Read-Only (R):** allows the file to be opened and read, but not edited or replaced.
- **Hidden (H):** the file is hidden from view. It won't show up in File Explorer (unless Explorer is set to show hidden files) and it won't appear in a folder listing when you use the `dir` command.
- **System (S):** marks the file as being part of the Windows system, giving it some protection from being deleted or replaced.

*See a file's
attributes*

To see the attributes of a file named `test.txt` for example, type the command **attrib test.txt**. You'll see something like this:

```
A R      test.txt
```

This tells you the file has the Archive and Read-only attributes (but not the Hidden or System attributes).

To change attributes, you use minus (hyphen) and plus signs with the appropriate letters. To remove the Read-only (R) attribute from a file you'd do this:

Add or remove attributes

attrib -r test.txt

To add the Read-only and Hidden attributes to a file, you'd do this:

attrib +r +h test.txt

CD – change directory

We met the cd command on page 9, and again on page 12, and you know almost everything you need to know about it. There's just one little thing I didn't cover.

Go to a different directory

I mentioned on page 9 that to change drives you type the drive letter and a colon, and that you can then use cd to navigate to the required folder. But cd actually lets you do both at the same time by adding the /d switch:

cd /d c:\users\rjy\documents

CHKDSK – check and repair your hard disk

If you find yourself in the situation that Windows won't start – perhaps because the PC has been switched off without giving Windows the chance to shut down – the first thing to try is the Startup Repair tool mentioned on page 8. With luck, that will do the job, but you won't always be lucky – something more serious may be wrong.

If Windows won't start, try Startup Repair

If Startup Repair doesn't work, the next thing to try is to open the Command Prompt and set about checking your hard disk for errors using the chkdsk command.

Failing that, try chkdsk

Chkdsk has a wide variety of switches and arguments that can be used, but we'll skip past those. You want to

know the command that's going to do the trick (if anything is) and it's this:

chkdsk c: /r /x

It may take a long time!

This tells the system to check your main hard disk (C:) for errors, and to repair any errors it finds. Note that this doesn't happen quickly: it may well take an hour or more to run, showing you its progress along the way, but it's likely to be time well spent!

COPY – copy a file (or files) elsewhere

If you want to copy one or more files elsewhere, this is the command that makes it possible. First, use the `cd` command to navigate to the directory containing the file you want to copy (and perhaps the `dir` command to determine its name).

Make a simple copy of a file

Now let's assume you want to make a copy of a file named `test.txt` in the same directory, giving it the name `test2.txt`:

copy test.txt test2.txt

Assuming there really is a file named `test.txt` in this directory, and there isn't yet a file named `test2.txt`, you'll see the note **1 file(s) copied**. Done!



If there was already a file named `test2.txt` in the directory, you'd instead see a note saying **Overwrite test2.txt? (Yes/No/All)**. Type **y** for Yes and press **Enter** if you do want to overwrite the existing file, or **n** for No if you don't.

To copy a file to a different directory, do this:

copy test.txt c:\users\rjy\documents

Copy to a different directory

Here you're giving the name of the file to be copied in the first argument, and the path to the directory in the second. By doing this, you'll end up with a file named

test.txt in that directory. If you wanted the copy to be named something else (such as test2.txt) you can specify that too:

copy test.txt c:\users\rjy\documents\test2.txt

One final thing you might want to do is to copy all the files from one directory to another – perhaps a directory you’ve created on a different drive (using the mkdir command, see below) in order to backup all those files. This command will do that:

copy *.* d:\backup\documents

The first argument (*.*) is an example of using something called ‘wildcards’ and it effectively means ‘all files and folders in this directory’. The second argument specifies the path to the directory where all these items should be copied.

Copy all files elsewhere

DEL – delete files

To delete a single file, use the command **del test.txt**. If you want to delete several files in one go, you can specify their names in a list: **del test.txt test2.txt test3.txt**. Note that anything you delete from the Command Prompt really is deleted: it doesn’t go into the Recycle Bin. To delete a folder, use **rmdir** (see page 18).

Delete one or more files

DIR – see the contents of a directory

We met the dir command on page 11: it displays a list of the files and folders in the current directory. There are various switches available (which you can see by typing **dir /?**) but you’ll rarely need to use them.

See what’s in the current directory

MKDIR – create a directory

The mkdir command is used to create a new directory (folder) inside the current one (or elsewhere if you provide the full path). To create a directory named

Make a new, empty folder

'test' inside the current one, just type **mkdir test**. To create a folder elsewhere, type the full path: **mkdir c:\users\rjy\test**.

REN – rename a file or directory

*Rename a file
or folder*

We met the **ren** command briefly earlier, and it's a simple command for renaming a file or directory (folder). It just needs two arguments: the item you want to rename and the new name you want to give it. For example, if there's a file named **test.txt** in the current folder and you want to rename it **test2.txt**, you do this:

ren test.txt test2.txt

If the file isn't in the current directory, you have to specify where it is in that first argument:

ren c:\users\rjy\documents\test.txt test2.txt

RMDIR – remove (delete) a directory

*Delete a directory
(and its contents)*

Whereas the **del** command (page 17) is used to delete files, you use **rmdir** to delete a folder. The question is, does that folder contain subfolders, and if it does, do you want to keep them? To delete a folder while keeping any subfolders it contains, you type this (assuming the unwanted directory is named 'test'):

rmdir test

That will delete all files from the directory, but leave the directory itself intact, still containing its subfolders and their files.

If you want to delete the directory and absolutely everything it contains, add the **/s** switch like this:

rmdir /s test