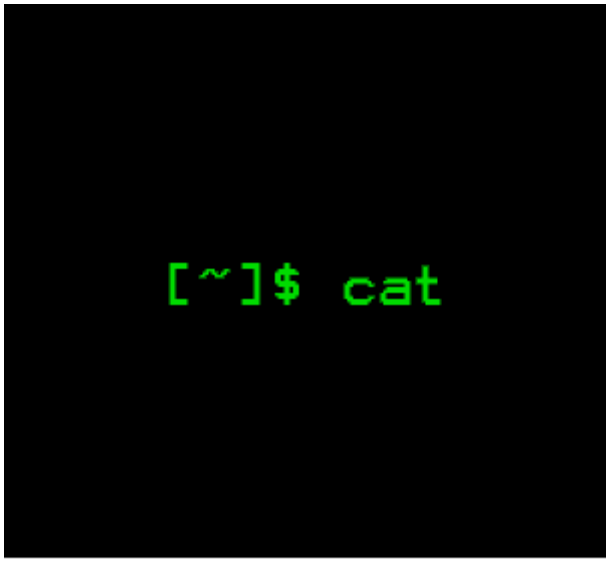
Linux and Unix cat command



http://www.computerhope.com

About cat

cat stands for "catenate." It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.

Overview

cat is one of the most commonly-used commands in Linux. It can be used to:

Display text files

Copy text files into a new document

Append the contents of a text file to the end of another text file, combining them

Displaying Text Files

The simplest way to use **cat** is to simply give it the name of a text file. It will display the contents of the text file on the screen. For instance:

```
cat mytext.txt
```

...will read the contents of **mytext.txt** and send them to standard output (your terminal screen). If **mytext.txt** is very long, they will zoom past and you will only see the last screen's worth of your document.

If you want to view the document page-by-page or scroll back and forth through the document, you can use a pager or viewer such as **pg**, **more**, or **less**.

If you specify more than one file name, **cat** will display those files one after the other, *catenating* their contents to standard output. So this command:

```
cat mytext.txt mytext2.txt
```

Will print the contents of those two text files as if they were a single file.

Copy A Text File

Normally you would copy a file with the **cp** command. You can use **cat** to make copies of text files in much the same way.

cat sends its output to stdout (standard output), which is usually the terminal screen. However, you can redirect this output to a file using the shell redirection symbol "**>**".

For instance, this command:

```
cat mytext.txt > newfile.txt
```

...will read the contents of **mytext.txt** and send them to standard output; instead of displaying the text, however, the shell will redirect the output to the file **newfile.txt**. If **newfile.txt** does not exist, it will be created.
If **newfile.txt** already exists, it will be overwritten and its previous contents will be lost, so be careful.

Similarly, you can catenate several files into your destination file. For instance:

```
cat mytext.txt mytext2.txt > newfile.txt
```

...will read the contents of **mytext.txt** and **mytext2.txt** and write the combined text to the file **newfile.txt**. Again, if **newfile.txt** does not already exist, it will be created; if it already exists, it will be overwritten.

Append A Text File's Contents To Another Text File

Instead of overwriting another file, you can also append a source text file to another using the redirection operator "**>>**".

For instance:

```
cat mytext.txt >> another-text-file.txt
```

...will read the contents of **mytext.txt**, and write them at the end of **another-text-file.txt**. If **another-text-file.txt** does not already exist, it will be created and the contents of **mytext.txt** will be written to the new file.

This works for multiple text files as well:

```
cat mytext.txt mytext2.txt >> another-text-file.txt
```

...will write the combined contents of **mytext.txt** and **mytext2.txt** to the end of **another-text-file.txt**.

Incorporating Standard Input Into cat Output

cat will read from standard input if you specify a hyphen ("**-**") as a file name. For example, if you have a file, **list.txt**, which contains the following text:

```
apples
oranges
butter
bread
```

...you could use the **echo** command to output text, **pipe** that output to **cat**, and instruct **cat** to catenate it with the file's contents, like this:

```
echo "My Shopping List" | cat - list.txt
```

...which would output the following text:

```
My Shopping List
apples
oranges
butter
bread
```

In short, **cat** is a simple but very useful tool for working with the data in text files on your system. This includes many important files like system logs and configuration files, and any other human-readable data stored in a file.

cat syntax

```
cat [OPTION]... [FILE]...
```

Options

These options are available on GNU **cat**, which is standard on most Linux distributions. If you are using a different Unix-like operating system (BSD, for example), some of these options may not be available; check your specific documentation for details.

-A, **--show-all**	Equivalent to **-vET**.
-b, **--number-nonblank**	Number non-empty output lines. This option overrides **-n**.
-e	Equivalent to **-vE**.
-E, **--show-ends**	Display "**$**" at end of each line.
-n, **--number**	Number all output lines.
-s, **--squeeze-blank**	Suppress repeated empty output lines.
-t	Equivalent to **-vT**.

-T, **--show-tabs**	Display TAB characters as **^I**.
-v, **--show-nonprinting**	Use **^** and **M-** notation, except for **LFD** and **TAB**.
--help	Display a help message, and exit.
--version	Output version information, and exit.

cat examples

```
cat file.txt
```

Read the contents of **file.txt** and display them on the screen.

```
cat file1.txt file2.txt
```

Reads the contents of **file1.txt** and **file2.txt**, and displays them in order on the terminal screen.

```
cat file.txt > newfile.txt
```

Read the contents of **file.txt** and write them to **newfile.txt**, overwriting anything **newfile.txt** previously contained. If **newfile.txt** does not exist, it will be created.

```
cat file.txt >> another-file.txt
```

Read the contents of **file.txt** and append them to the end of **another-file.txt**. If **another-file.txt** does not exist, it will be created.

```
cat -s file.txt
```

Display the contents of **file.txt**, omitting any repeated blank lines.

Related commands

cp — Copy files and directories.
ed — A simple text editor.
less — Scrolling text viewer.
more — Display text one screen at a time.
pico — A simple text editor.
pg — Browse page by page through text files.
tac — Output the contents of files in reverse order.
tee — Route a file's contents to multiple outputs.
touch — Update the timestamp of a file or directory.