# Manipulating Text

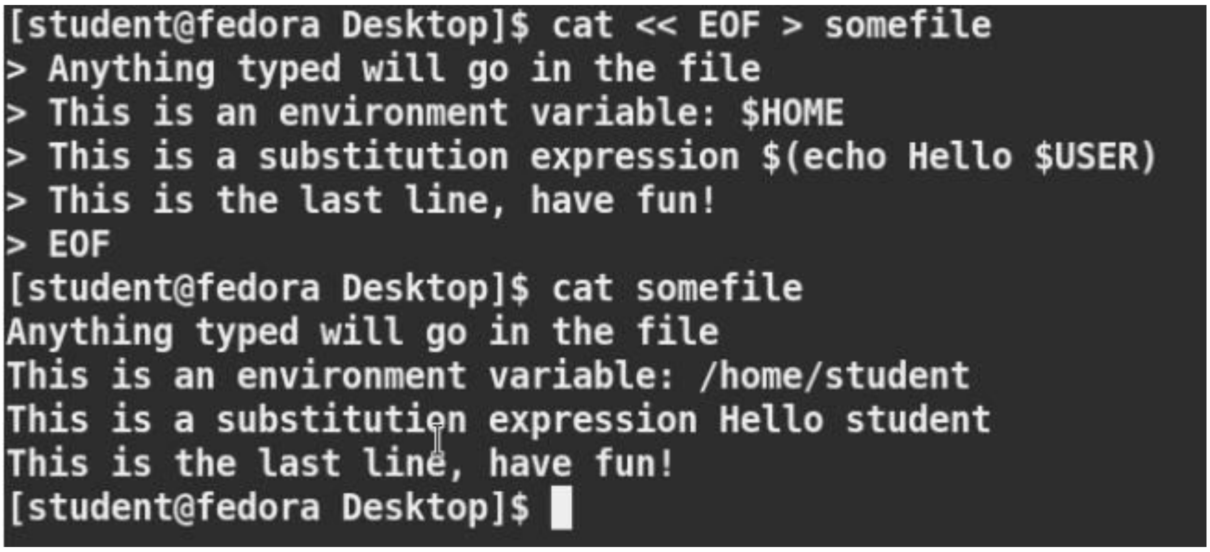
## cat:

1. **cat** is short for **concatenate** and is one of the most frequently used Linux command line utilities.
2. It is often used to read and print files, as well as for simply viewing file contents.
3. To view a file, use the following command
```
$ cat <filename>
```
4. For example, **cat readme.txt** will display the contents of readme.txt on the terminal.
5. However, the main purpose of **cat** is often to combine (concatenate) multiple files together.
6. You can perform the actions listed in the table using **cat**.
7. The **tac** command (**cat** spelled backwards) prints the lines of a file in reverse order.
8. Each line remains the same, but the order of lines is inverted.
9. The syntax of **tac** is exactly the same as for **cat**, as in the below
```
$ tac file
$ tac file1 file2 > newfile
```

| Command | Usage |
|---|---|
| `cat file1 file2` | Concatenate multiple files and display the output; i.e. the entire content of the first file is followed by that of the second file |
| `cat file1 file2 > newfile` | Combine multiple files and save the output into a new file |
| `cat file >> existingfile` | Append a file to the end of an existing file |
| `cat > file` | Any subsequent lines typed will go into the file, until **CTRL-D** is typed |
| `cat >> file` | Any subsequent lines are appended to the file, until **CTRL-D** is typed |

## Using cat Interactively:

1. **cat** can be used to read from standard input (such as the terminal window) if no files are specified.
2. You can use the **>** operator to create and add lines into a new file, and the **>>** operator to append lines (or files) to an existing file.
3. To create a new file, at the command prompt type **cat > <filename>** and press the **Enter** key.
4. This command creates a new file and waits for the user to edit/enter the text.
5. After you finish typing the required text, press **CTRL-D** at the beginning of the next line to save and exit the editing.
6. Another way to create a file at the terminal is **cat > <filename> << EOF**. A new file is created and you can type the required input
7. To exit, enter **EOF** at the beginning of a line.
8. Note that **EOF** is case sensitive. One can also use another word, such as **STOP**.



```
[student@fedora Desktop]$ cat << EOF > somefile
> Anything typed will go in the file
> This is an environment variable: $HOME
> This is a substitution expression $(echo Hello $USER)
> This is the last line, have fun!
> EOF
[student@fedora Desktop]$ cat somefile
Anything typed will go in the file
This is an environment variable: /home/student
This is a substitution expression Hello student
This is the last line, have fun!
[student@fedora Desktop]$ █
```

## echo:

1. **echo** simply displays (echoes) text. It is used simply, as in:
   $ echo string
2. echo can be used to display a string on standard output (i.e. the terminal) or to place in a new file (using the **>** operator) or append to an already existing file (using the **>>** operator).
3. The **–e** option, along with the following switches, is used to enable special character sequences, such as the newline character or horizontal tab:
   \n represents newline
   \t represents horizontal tab.
4. **echo** is particularly useful for viewing the values of environment variables (built-in shell variables).
5. For example, **echo $USERNAME** will print the name of the user who has logged into the current terminal.
6. The following table lists echo commands and their usage:

| Command | Usage |
|---|---|
| `echo string > newfile` | The specified string is placed in a new file |
| `echo string >> existingfile` | The specified string is appended to the end of an already existing file |
| `echo $variable` | The contents of the specified environment variable are displayed |

## Working with Large Files:

1. If we try to open a large file in editor mode it will cause issues due to high memory utilization, as an editor will usually try to read the whole file into memory first.
2. However, one can use **less** to view the contents of such a large file, scrolling up and down page by page, without the system having to place the entire file in memory before starting. This is much faster than using a text editor.
3. Viewing somefile can be done by typing either of the two following commands:
   $ less somefile
   $ cat somefile | less

4. By default, man pages are sent through the less command.

## head:

1. **head** reads the first few lines of each named file (10 by default) and displays it on standard output. You can give a different number of lines in an option.
2. For example, if you want to print the first 5 lines from /etc/default/grub, use the following command:
```
$ head –n 5 /etc/default/grub
```
**Or**
```
$ head -5 /etc/default/grub
```

## tail:

1. **tail** prints the last few lines of each named file and displays it on standard output.
2. By default, it displays the last 10 lines. You can give a different number of lines as an option.
3. tail is especially useful when you are troubleshooting any issue using log files, as you probably want to see the most recent lines of output.
4. For example, to display the last 15 lines of somefile.log, use the following command:
```
$ tail -n 15 somefile.log
```
**Or**
```
$ tail -15 somefile.log
```
5. To continually monitor new output in a growing log file use the below command
6. The below command will continuously display any new lines of output in somefile.log as soon as they appear. Thus, it enables you to monitor any current activity that is being reported and recorded.
```
$ tail -f somefile.log
```

## Viewing Compressed Files:

1. When working with compressed files, many standard commands cannot be used directly.
2. For many commonly-used file and text manipulation programs, there is also a version especially designed to work directly with compressed files.
3. These associated utilities have the letter "**z**" prefixed to their name. For example, we have utility programs such as **zcat**, **zless**, **zdiff** and **zgrep**.
4. Here is a table listing some **z** family commands:

| Command | Description |
|---|---|
| `$ zcat compressed-file.txt.gz` | To view a compressed file |
| `$ zless somefile.gz`<br>or<br>`$ zmore somefile.gz` | To page through a compressed file |
| `$ zgrep -i less somefile.gz` | To search inside a compressed file |
| `$ zdiff file1.txt.gz file2.txt.gz` | To compare two compressed files |

5. Note that if you run **zless** on an uncompressed file, it will still work and ignore the decompression stage.
6. There are also equivalent utility programs for other compression methods besides **gzip**, for example, we have **bzcat** and **bzless** associated with **bzip2**, and **xzcat** and **xzless** associated with **xz**.