Basic Terminal Usage

Launch a terminal from your desktop's application menu and you will see the bash shell. There are other shells, but most Linux distributions use bash by default.

You can launch a program by typing its name at the prompt. Everything you launch here – from graphical applications like Firefox to command-line utilities – is a program. (Bash actually has a few built-in commands for basic file management and such, but those function like programs, too.) Unlike on Windows, you don't have to type the full path to a program to launch it. For example, let's say you wanted to open Firefox. On Windows, you'd need to type the full path to Firefox's .exe file. On Linux, you can just type:

firefox

as long as the firefox application is in the path – often it will be in /usr/bin/

There is one quirk, though. If you want to run an application in the current directory, you have to tell bash to do so. For example:

./firefox

(in Linux . is the current directory, .. is the directory above the current one)

To work with files and directories, you will need to know a few basic commands:

- **pwd –** print the full path of the current working directory. It stands for "Print Working Directory"
- **cd** – change directory. For example, you might cd /path/to/somewhere/ but you will probably do cd ~ quite a lot. That **~** represents your home directory (that's /home/you), which is the terminal's default directory. To change to another directory, you can use the **cd** command. For example **cd /** would change to the root directory, **cd Downloads** would change to the Downloads directory inside the current directory (so this only opens your Downloads directory if the terminal is in your home directory), **cd /home/you/Downloads** would change to your Downloads directory from anywhere in the system, **cd ~** would change to your home directory, and **cd ..** would go up a directory.
- **ls** – The **ls** command lists the files in the current directory. You can also list files recursively — that is, list all files in directories inside the current directory — with **ls -R**.
- **ls -a** list all files, including hidden files
- **ls -l** use a long listing format list of files together with file permissions and modification dates
- **touch** – change file date/time or create empty files. The **touch** command creates an empty file if that file doesn't already exist. For example, **touch example** creates an empty file named "example" in the current directory. Touch is also used to update access and modification time/date of existing files.

- **mkdir** – The **mkdir** command makes a new directory. **mkdir example** would create a new directory named example in the current directory, while **mkdir /home/you/Downloads/test** would create a new directory named **test** in your Downloads directory. You can make deeper nested directories with **mkdir -p** for example **mkdir -p /home/you/this/is/a/deep/directory/tree**
- **rm** – The **rm** command removes a file. For example, **rm example** removes the file named example in the current directory and **rm /home/you/Downloads/example** removes the file named **example** in the Downloads directory.
- **rmdir** – Remove Directories - the **rmdir** command removes an empty directory. **rmdir directory** would delete the directory named “directory” in the current directory.
  If the directory isn’t empty, you can use a recursive rm command to remove the directory and all files in it. **rm -r directory** would delete the directory named “directory” and all files in it. **BE VERY CAREFUL! rm -r is a dangerous command that could easily delete a lot of important files, so be careful when using it. It won’t ask for confirmation!**
- **cp** – The **cp** command copies a file from one location to another. For example, **cp example /home/you/Downloads** copies the file named **example** in the current directory to /home/you/Downloads. You can use **cp -r** to recurse subdirectories.
- **mv** – The **mv** command moves a file from one location to another. It works exactly like the cp command above, but moves the file instead of creating a copy. mv can also be used to rename files. For example**, mv original renamed** moves a file named **original** in the current directory to a file named **renamed** in the current directory, effectively renaming it.
- **cat [text file]** - display all the contents of a text file at the terminal.
  For large files, we can use cat [text file] | more and cat [text file] | less, the same way we pipe the output of ls or locate.
  We can also use head [text file] to only show the first ten lines, and tail [text file] to display the last ten lines. The actual number of lines can be adjusted with switches.
- **less/more** If we run "ls" on a directory with 1,000 files we will get too many results and they will scroll off the screen. In this case, we can use a pipe with the vertical bar "|" symbol and **more** or **less**. We will get the results page by page, and we can reveal the next pages by pressing space. We quit by pressing "q".
- **nano [text file]** - nano is a relatively user-friendly terminal text editor. We can also run nano with no parameters, to create a new text file. Just remember that the commands on the bottom all work with Ctrl, e.g. we press Ctrl+X to exit, Ctrl+G to get help, etc.
- `grep` command is used to search text. It searches the given file for lines containing a match to the given strings or words.

  ```
  grep <word> <filename>
  ```

- **sudo and apt-get** - An example of a command you might use regularly from the command line is apt-get which is used to install and remove software within Debian and Ubuntu based distributions. Normal users aren’t permitted to install software or change system settings, which is why sudo is used.

sudo ("superuser do", or "switch user do") allows a user with proper permissions to execute a command as another user, such as the superuser. The Sudo command allows you to run a command as *any* user, with the default generally being the root.

- To do a "dry run" of a procedure in order to get an idea of what an apt-get action will do, you can pass the "-s" flag for "simulate":

```
sudo apt-get install -s aha
```

HINTS

Tab Completion

Tab completion is an essential trick. It's a great time saver and it's also useful if you're not sure of a file or command's exact name.

For example, let's say you have a file named "really long file name" in the current directory and you want to delete it. You could type the entire file name, but you'd have to escape the space characters properly (in other words, add the \ character before each space) and might make a mistake. If you type **rm r** and press Tab, Bash will automatically fill the file's name in for you.

Of course, if you have multiple files in the current directory that begin with the letter r, Bash won't know which one you want. Let's say you have another file named "really very long file name" in the current directory. When you hit Tab, Bash will fill in the "really\ " part, since the files both begin with that. After it does, press Tab again and you'll see a list of matching file names. Type the next character of the file you want, press Tab again and bash will complete more of the name, up to any other multiple matches.

Pipes

Pipes allow you to send the output of a command to another command. In the UNIX philosophy, each program is a small utility that do one thing well. Like Lego, you can put them together in lots of different ways to achieve what you want. This is one of the most powerful features of Linux. For example, the **ls** command lists the files in the current directory and the **grep** command searches its input for a specified term.

Combine these with pipes (the **|** character) and you can search for a file in the current directory. The following command searches for the word "word":

```
ls | grep word
```

Wild Cards

The ***** character – that is, the asterisk – is a wild card that can match anything. For example, if we wanted to delete both "really long file name" and "really very long file name" from the current directory, we could run the following command:

```
rm really*name
```

This command deletes all files with file names beginning with “really” and ending with “name.” If you ran **rm *** instead, you’d delete every file in the current directory, so be careful.

The other common wildcard is the ? character, which replaces just a single character in a string. So, for example, d?g would match dog and dig.

Output Redirection

The > character redirects a command’s output to a file instead of another command. For example, the following line runs the **ls** command to list the files in the current directory and, instead of printing that list to the terminal, it prints the list to a file named “file1” in the current directory:

```
ls > file1
```

Command History

Bash remembers a history of the commands you type into it. You can use the up and down arrow keys to scroll through commands you’ve recently used. The **history** command prints a list of these commands, so you can pipe it to grep to search for commands you’ve used recently.

~ and . and ..

The ~ character – also known as the tilde – represents the current user’s home directory. So, instead of typing **cd /home/name** to go to your home directory, you can type **cd ~** instead. This also works with relative paths – **cd ~/Desktop** would switch to the current user’s desktop.

Similarly, the **.** represents the current directory and the **..** represents the directory above the current directory. So, **cd ..** goes up a directory. These also work with relative paths – if you’re in your Desktop folder and want to go to the Documents folder, which is in the same directory as the Desktop folder, you can use the **cd ../Documents** command.

Open New Terminal

To open a new terminal from the GUI, the shortcut **Ctrl+Alt+T** will work on most distributions and desktop environments.

Extra information about Linux commands

We can learn more about any of the Linux commands with **[command] --help** and **man [command]**.

[command] --help will show the usage of the command, and the available options and switches.

Sources:
https://www.howtogeek.com/140679/beginner-geek-how-to-start-using-the-linux-terminal/
https://www.howtogeek.com/110150/become-a-linux-terminal-power-user-with-these-8-tricks/