1. touch = The touch command can be used not only to create empty files, but also to update the access and modification times of existing files.
2. file = You can use file [filename] to determine a file’s type (this will come in handy before launching your preferred text editor to edit it).
3. rm = rm [filename] is used to delete the file.
4. mkdir = As for directories, you can create directories inside existing paths with mkdir [directory] or create a full path with mkdir -p [/full/path/to/directory].
5. cd = cd command is used to change the current working directory.
6. rmdir = It is used to removing the directory. When it comes to removing directories, you need to make sure that they’re empty before issuing the rmdir [directory] command
7. rm -rf [directory] = This option will force remove recursively the [directory] and all its contents – so use it at your own risk.

|  |  |
| --- | --- |
| **Redirection Operator** | **Effect** |
| **>** | Redirects standard output to a file containing standard output. If the destination file exists, it will be overwritten. |
| **>>** | Appends standard output to a file. |
| **2>** | Redirects standard error to a file containing standard output. If the destination file exists, it will be overwritten. |
| **2>>** | Appends standard error to the existing file. |
| **&>** | Redirects both standard output and standard error to a file; if the specified file exists, it will be overwritten. |
| **<** | Uses the specified file as standard input. |
| **< >** | The specified file is used for both standard input and standard output. |

1. **Redirection vs Pipelining** = As opposed to redirection, pipelining is performed by adding a vertical bar (|) after a command and before another one.

Remember:

**Redirection** is used to send the output of a command to a file, or to send a file as input to a command.

**Pipelining** is used to send the output of a command to another command as input.

1. grep = The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).**EX:**

ps -ef | grep apache | grep -v grep

1. **Archiving, Compressing, Unpacking, and Uncompressing Files**

If you need to transport, backup, or send via email a group of files, you will use an archiving (or grouping) tool such as [tar](https://www.tecmint.com/18-tar-command-examples-in-linux/), typically used with a compression utility like **gzip**, **bzip2**, or **xz**.

Your choice of a compression tool will be likely defined by the compression speed and rate of each one. Of these three compression tools, **gzip** is the oldest and provides the least compression, **bzip2** provides improved compression, and **xz** is the newest and provides the best compression. Typically, files compressed with these utilities have **.gz**, **.bz2**, or **.xz** extensions, respectively.

|  |  |  |
| --- | --- | --- |
| **Command** | **Abbreviation** | **Description** |
| –create | c | Creates a tar archive |
| –concatenate | A | Appends tar files to an archive |
| –append | r | Appends non-tar files to an archive |
| –update | u | Appends files that are newer than those in an archive |
| –diff or –compare | d | Compares an archive to files on disk |
| –list | t | Lists the contents of a tarball |
| –extract or –get | x | Extracts files from an archive |

|  |  |  |
| --- | --- | --- |
| **Operation modifier** | **Abbreviation** | **Description** |
| —directory dir | C | Changes to directory dir before performing operations |
| —same-permissions and —same-owner | p | Preserves permissions and ownership information, respectively. |
| –verbose | v | Lists all files as they are read or extracted; if combined with –list, it also displays file sizes, ownership, and timestamps |
| —exclude file | — | Excludes file from the archive. In this case, file can be an actual file or a pattern. |
| —gzip or —gunzip | z | Compresses an archive through gzip |
| –bzip2 | j | Compresses an archive through bzip2 |
| –xz | J | Compresses an archive through xz |

1. **Create Hard and Soft Links**

In Linux, there are two types of links to files: **hard links** and **soft** (aka symbolic) links. Since a hard link represents another name for an existing file and is identified by the same **inode**, it then points to the actual data, as opposed to symbolic links, which point to filenames instead.

In addition, hard links do not occupy space on disk, while symbolic links do take a small amount of space to store the text of the link itself. The downside of hard links is that they can only be used to reference files within the filesystem where they are located because inodes are unique inside a filesystem. Symbolic links save the day, in that they point to another file or directory by name rather than by inode, and therefore can cross filesystem boundaries.

The basic syntax to create links is similar in both cases:

# ln TARGET LINK\_NAME # Hard link named LINK\_NAME to file named TARGET

# ln -s TARGET LINK\_NAME # Soft link named LINK\_NAME to file named T