“In Linux, everything is a File”.

That is in fact true although it is just a generalization concept, in Unix and its derivatives such as Linux, everything is considered as a file. If something is not a file, then it must be running as a process on the system.

To understand this, take for example the amount of space on your root (/) directory is always consumed by different types of Linux files. When you create a file or transfer a file to your system, it occupies some space on the physical disk and it is considered to be in a specific format (file type).

And also the Linux system does not differentiate between files and directories, but directories do one important job, that is store other files in groups in a hierarchy for easy location. All your hardware components are represented as files and the system communicates with them using these files.

The idea is an important description of a great property of Linux, where input/output resources such as your documents, directories (folders in Mac OS X and Windows), keyboard, monitor, hard-drives, removable media, printers, modems, virtual terminals and also inter-process and network communication are streams of bytes defined by file system space.

A notable advantage of everything being a file is that the same set of Linux tools, utilities and APIs can be used on the above input/output resources.

Although everything in Linux is a file, there are certain special files that are more than just a file for example sockets and named pipes.

What are the different types of files in Linux?

In Linux there are basically three types of files:

- Ordinary/Regular files
- Special files
- Directories

Ordinary/Regular Files

These are files data contain text, data or program instructions and they are the most common type of files you can expect to find on a Linux system and they include:

- Readable files
- Binary files
- Image files
- Compressed files and so on.

Special Files

Special files include the following:

Block files : These are device files that provide buffered access to system hardware components. They provide a method of communication with device drivers through the file system.

One important aspect about block files is that they can transfer a large block of data and information at a given time.

Listing block files sockets in a directory:

```
ls -l /dev | grep "^b"
```

Sample Output

```
brw-rw----  1 root  disk      7,   0 Jan  4 19:50 loop0
brw-rw----  1 root  disk      7,   1 Dec 18 05:00 loop1
brw-rw----  1 root  disk      7,   2 Jan 11 20:16 loop2
brw-rw----  1 root  disk      7,   3 Dec  4 19:05 loop3
brw-rw----  1 root  disk      7,   4 Jan 11 20:16 loop4
brw-rw----  1 root  disk      7,   5 Dec 17 04:25 loop5
brw-rw----  1 root  disk      7,   6 Oct 27 07:10 loop6
brw-rw----  1 root  disk      7,   7 Jan  4 19:51 loop7
brw-rw----  1 root  disk      7,   8 Oct 19 18:50 loop8
brw-rw----  1 root  disk      8,   0 Aug 24 17:30 sda
brw-rw----  1 root  disk      8,   1 Aug 24 17:29 sda1
brw-rw----  1 root  disk      8,   2 Aug 24 17:29 sda2
brw-rw----+ 1 root  cdrom    11,   0 Aug 24 17:29 sr0
```

Character files : These are also device files that provide unbuffered serial access to system hardware components. They work by providing a way of communication with devices by transferring data one character at a time.

Listing character files sockets in a directory:

```
ls -l /dev | grep "^c"
```

Symbolic link files : A symbolic link is a reference to another file on the system. Therefore, symbolic link files are files that point to other files, and they can either be directories or regular files.

Listing symbolic link sockets in a directory:

```
ls -l /dev/ | grep "^l"
```

Sample Output

```
lrwxrwxrwx  1 root  root           3 Aug 24 17:29 cdrom -> sr0
lrwxrwxrwx  1 root  root           3 Aug 24 17:29 cdrw -> sr0
lrwxrwxrwx  1 root  root          11 Aug 24 17:29 core -> /proc/kcore
lrwxrwxrwx  1 root  root           3 Aug 24 17:29 dvd -> sr0
lrwxrwxrwx  1 root  root          13 Aug 24 17:29 fd -> /proc/self/fd
lrwxrwxrwx  1 root  root          12 Aug 24 17:29 initctl -> /run/ini
tctl
lrwxrwxrwx  1 root  root          28 Aug 24 17:29 log -> /run/systemd
/journal/dev-log
lrwxrwxrwx  1 root  root           4 Aug 24 17:29 rtc -> rtc0
lrwxrwxrwx  1 root  root          15 Aug 24 17:29 stderr -> /proc/sel
f/fd/2
lrwxrwxrwx  1 root  root          15 Aug 24 17:29 stdin -> /proc/self
/fd/0
lrwxrwxrwx  1 root  root          15 Aug 24 17:29 stdout -> /proc/sel
f/fd/1
```

You can make symbolic links using the ln utility in Linux as in the example below.

```
touch file1.txt
ln -s file1.txt /home/efkan/file1.txt [create symbolic link]
ls -l /home/efkan/ | grep "^l" [List symbolic links]
```

In the above example, I created a file called `file1.txt` in `/tmp` directory, then created the symbolic link, `/home/efkan/file1.txt` to point to `/tmp/file1.txt`.

Pipes or Named pipes : These are files that allow inter-process communication by connecting the output of one process to the input of another.
A named pipe is actually a file that is used by two process to communicate with each and it acts as a Linux pipe.
Listing pipes sockets in a directory:

```
ls -l | grep "^p"
```

Sample Output

```
prw-rw-r-- 1 efkan efkan    0 May 18 17:47 pipe1
prw-rw-r-- 1 efkan efkan    0 May 18 17:47 pipe2
prw-rw-r-- 1 efkan efkan    0 May 18 17:47 pipe3
prw-rw-r-- 1 efkan efkan    0 May 18 17:47 pipe4
prw-rw-r-- 1 efkan efkan    0 May 18 17:47 pipe5
```

You can use the `mkfifo` utility to create a named pipe in Linux as follows.

```
mkfifo pipe1
echo "This is named pipe1" > pipe1
```

In the above example, I created a named pipe called pipe1, then I passed some data to it using the echo command, after that the shell became un-interactive while processing the input.
Then I opened another shell and run the another command to print out what was passed to pipe.

```
while read line ;do echo "This was passed-'$line' "; done<pipe1
```

Socket files : These are files that provide a means of inter-process communication, but they can transfer data and information between process running on different environments.
This means that sockets provide data and information transfer between process running on different machines on a network.
An example to show the work of sockets would be a web browser making a connection to a web server.

```
ls -l /dev/ | grep "^s"
```

Sample Output

```
srw-rw-rw- 1 root root 0 May 18 10:26 log
```

This is an example of a socket create in C by using the socket() system call.

```
int socket_desc= socket(AF_INET, SOCK_STREAM, 0 );
```

In the above:

- AF_INET is the address family(IPv4)
- SOCK_STREAM is the type (connection is TCP protocol oriented)
- 0 is the protocol(IP Protocol)

To refer to the socket file, use the socket_desc, which is the same as the file descriptor, and use read() and write() system calls to read and write from the socket respectively.

Directories

These are special files that store both ordinary and other special files and they are organized on the Linux file system in a hierarchy starting from the root (/) directory.
Listing sockets in a directory:

```
ls -l / | grep "^d"
```

Sample Output

```
drwxr-xr-x   2 root root  4096 May  5 15:49 bin
drwxr-xr-x   4 root root  4096 May  5 15:58 boot
drwxr-xr-x   2 root root  4096 Apr 11  2015 cdrom
drwxr-xr-x  17 root root  4400 May 18 10:27 dev
drwxr-xr-x 168 root root 12288 May 18 10:28 etc
drwxr-xr-x   3 root root  4096 Apr 11  2015 home
drwxr-xr-x  25 root root  4096 May  5 15:44 lib
drwxr-xr-x   2 root root  4096 May  5 15:44 lib64
drwx------   2 root root 16384 Apr 11  2015 lost+found
drwxr-xr-x   3 root root  4096 Apr 10  2015 media
drwxr-xr-x   3 root root  4096 Feb 23 17:54 mnt
drwxr-xr-x  16 root root  4096 Apr 30 16:01 opt
dr-xr-xr-x 223 root root     0 May 18 15:54 proc
drwx------  19 root root  4096 Apr  9 11:12 root
drwxr-xr-x  27 root root   920 May 18 10:54 run
drwxr-xr-x   2 root root 12288 May  5 15:57 sbin
drwxr-xr-x   2 root root  4096 Dec  1  2014 srv
dr-xr-xr-x  13 root root     0 May 18 15:54 sys
drwxrwxrwt  13 root root  4096 May 18 17:55 tmp
drwxr-xr-x  11 root root  4096 Mar 31 16:00 usr
drwxr-xr-x  12 root root  4096 Nov 12  2015 var
```

You can make a directory using the mkdir command.

```
mkdir -m 1666 devops.com
mkdir -m 1666 iba.devops.com
mkdir -m 1775 demo.com
```

Summary

You should now be having a clear understanding of why everything in Linux is a file and the different types of files that can exit on your Linux system.