grep command:

Grep is an essential Linux and Unix command. It is used to search text and strings in a given file. In other words, grep command searches the given file for lines containing a match to the given strings or words.

```
grep foo index.txt

grep foo -w index.txt

grep foo -i index.txt

grep foo -c index.txt

grep foo -c -i index.txt

grep foo -R
```

cat Command:

Cat(concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate files.

```
cat press enter

Hello

Hello
```

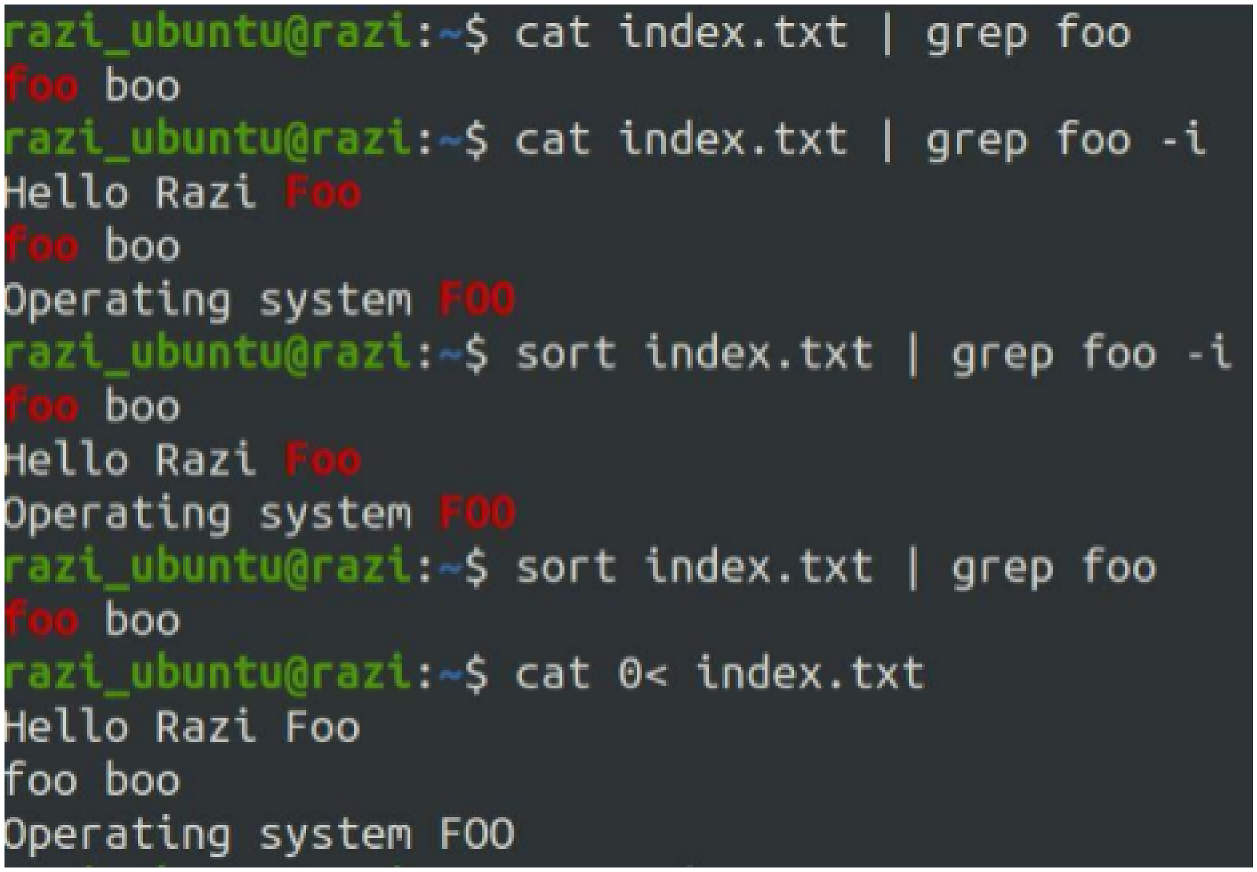
Sort Command:

The sort command sorts the contents of a file, in numeric or alphabetic order, and prints the results to standard output (usually the terminal screen). The original file is unaffected. **It comes in different flavors. You can read about it by typing man sort command in your terminal.**

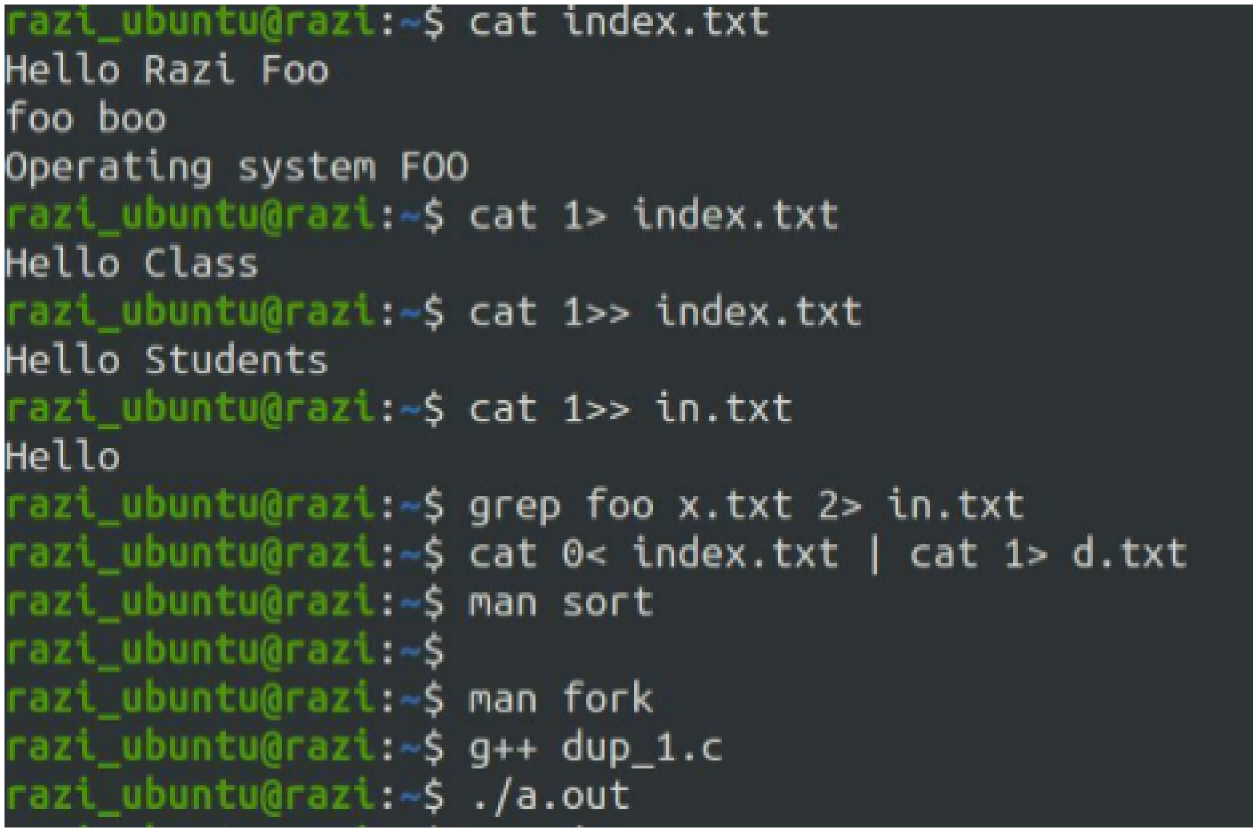
pipe:

Pipes can also be used on the command line to connect the standard input of one process to the standard input of another. This is done by using the pipe operator which is | and the syntax is as follows:

```
cmd1 | cmd2 | ... | cmdN
```



```
razi_ubuntu@razi:~$ cat index.txt | grep foo
foo boo
razi_ubuntu@razi:~$ cat index.txt | grep foo -i
Hello Razi Foo
foo boo
Operating system FOO
razi_ubuntu@razi:~$ sort index.txt | grep foo -i
foo boo
Hello Razi Foo
Operating system FOO
razi_ubuntu@razi:~$ sort index.txt | grep foo
foo boo
razi_ubuntu@razi:~$ cat 0< index.txt
Hello Razi Foo
foo boo
Operating system FOO
```



```
razi_ubuntu@razi:~$ cat index.txt
Hello Razi Foo
foo boo
Operating system FOO
razi_ubuntu@razi:~$ cat 1> index.txt
Hello Class
razi_ubuntu@razi:~$ cat 1>> index.txt
Hello Students
razi_ubuntu@razi:~$ cat 1>> in.txt
Hello
razi_ubuntu@razi:~$ grep foo x.txt 2> in.txt
razi_ubuntu@razi:~$ cat 0< index.txt | cat 1> d.txt
razi_ubuntu@razi:~$ man sort
razi_ubuntu@razi:~$
razi_ubuntu@razi:~$ man fork
razi_ubuntu@razi:~$ g++ dup_1.c
razi_ubuntu@razi:~$ ./a.out
```

dup():

The dup() system call creates a copy of a file descriptor.

• It uses the lowest-numbered unused descriptor for the new descriptor. • If the copy is successfully created, then the original and copy file descriptors may be used interchangeably.
• They both refer to the same open file description and thus share file offset and file status flags.

```
int dup(int oldfd);
oldfd: old file descriptor whose copy is to be created.
```

dup2():

The dup2() system call is similar to dup() but the basic difference between them is that instead of using the lowest-numbered unused file descriptor, it uses the descriptor number specified by the user.

```
int dup2(int oldfd, int newfd);

oldfd: old file descriptor
 newfd new file descriptor which is used by dup2() to create a copy.
```

• Include the header file unistd.h for using dup() and dup2() system call. • If the descriptor newfd was previously open, it is silently closed before being reused.
• If oldfd is not a valid file descriptor, then the call fails, and newfd is not closed.
• If oldfd is a valid file descriptor, and newfd has the same value as oldfd, then dup2() does
• nothing, and returns newfd.