wc Command

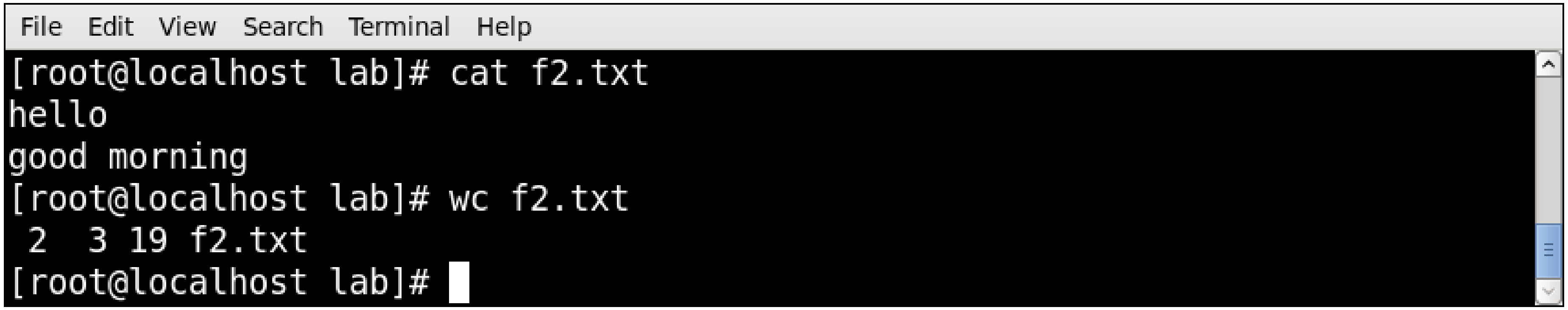
- It s used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments.

- **Syntax :**

```
wc [options] filenames
```

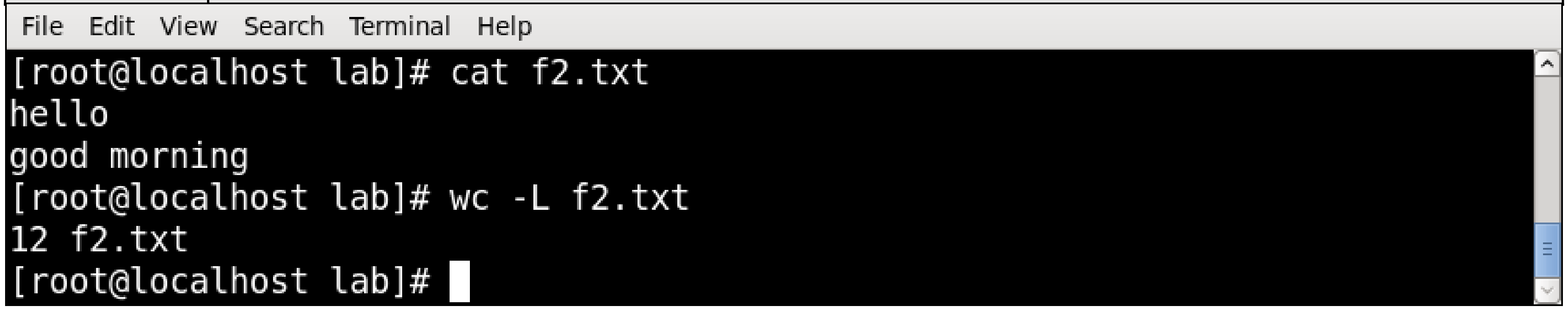
Option	Use
wc -l	Prints the number of lines in a file
wc -w	Prints the number of words in a file
wc -c	Displays the count of bytes in a file
wc -L	Prints only the length of the longest line in a file

wc Command Example



```
[root@localhost lab]# cat f2.txt
hello
good morning
[root@localhost lab]# wc f2.txt
 2  3 19 f2.txt
[root@localhost lab]# 
```

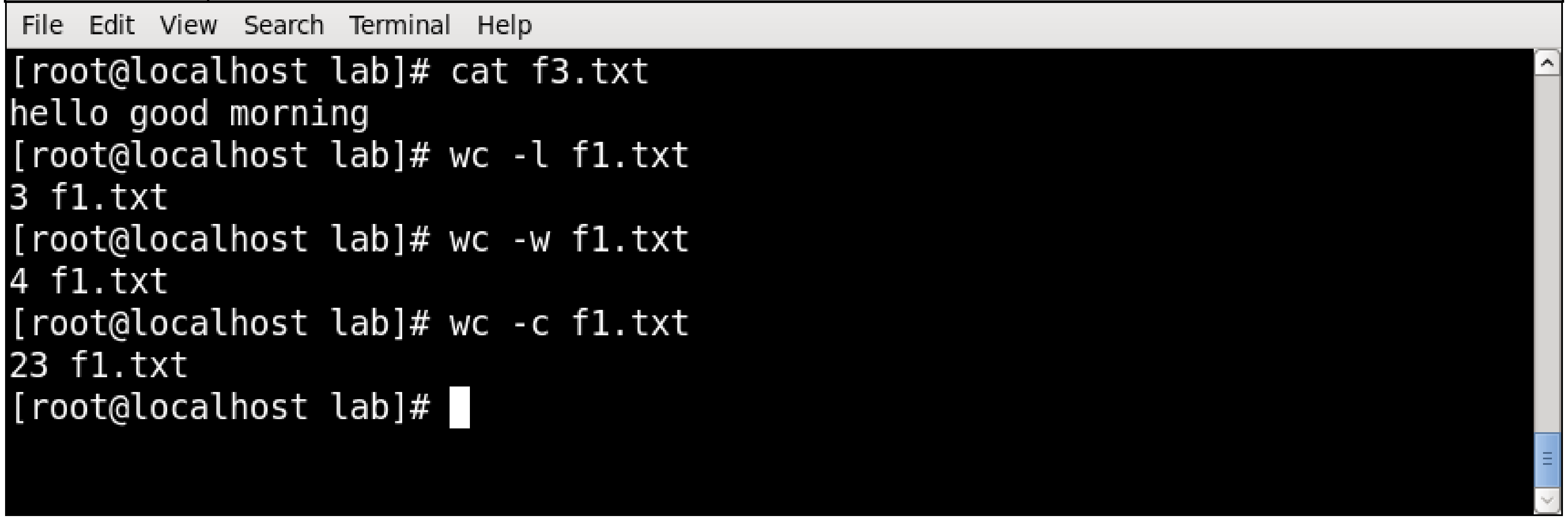
wc -L	**Prints only the length of the longest line in a file**



```
[root@localhost lab]# cat f2.txt
hello
good morning
[root@localhost lab]# wc -L f2.txt
12 f2.txt
[root@localhost lab]# 
```

wc Command Example

wc -l	**Prints the number of lines in a file**
wc -w	**Prints the number of words in a file**
wc -c	**Displays the count of bytes in a file**



```
[root@localhost lab]# cat f3.txt
hello good morning
[root@localhost lab]# wc -l f1.txt
3 f1.txt
[root@localhost lab]# wc -w f1.txt
4 f1.txt
[root@localhost lab]# wc -c f1.txt
23 f1.txt
[root@localhost lab]# 
```

ln Command

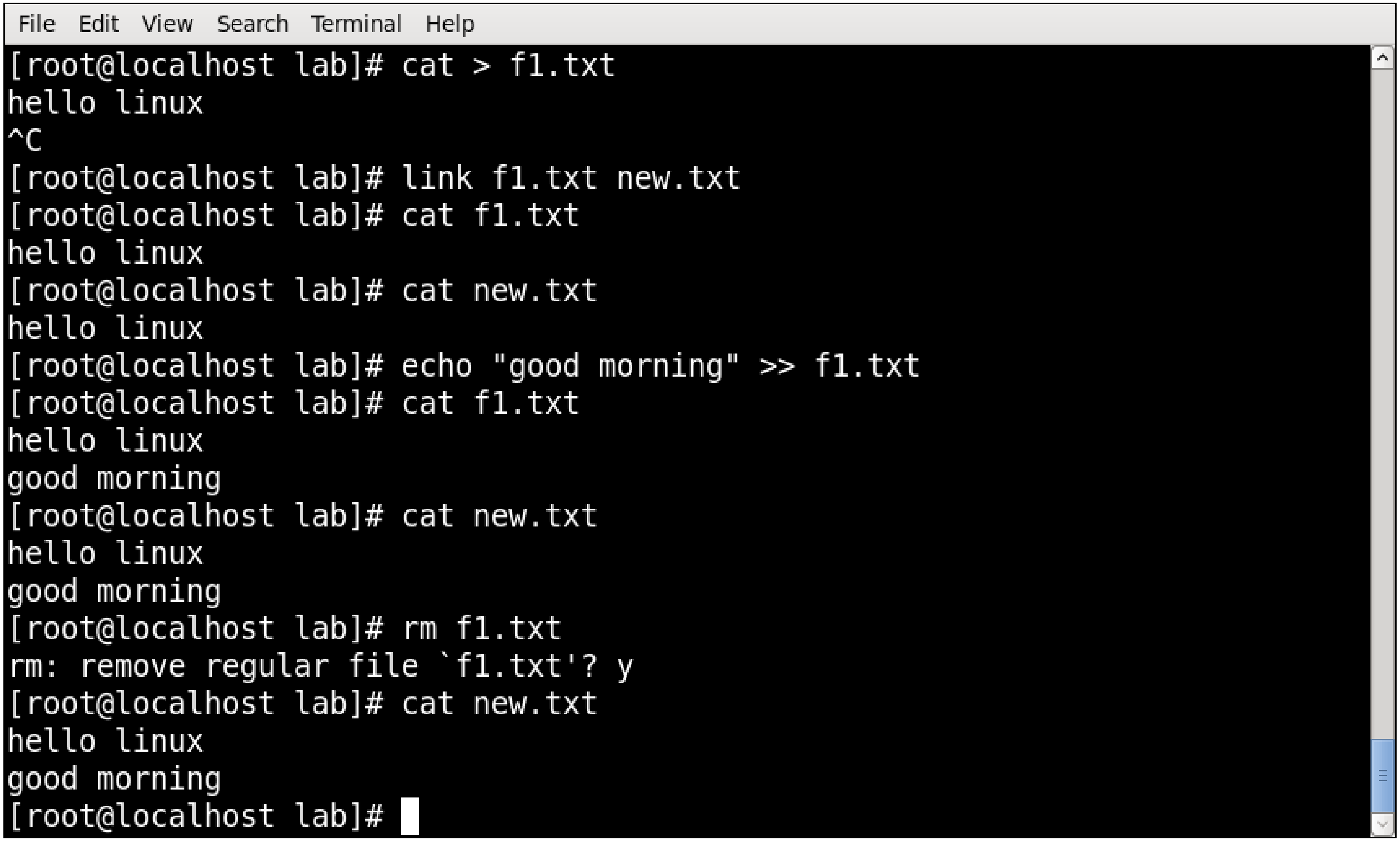
- **ln** creates links between files.
- ln creates hard links by default, or symbolic links if the -s (--symbolic) option is specified. When creating hard links, each TARGET must exist.

- **Syntax :**

  ln [OPTION]... [-T] TARGET LINK_NAME

Option	Use
ln -f	If the destination file or files already exist, overwrite them
ln -i	Prompt the user before overwriting destination files
ln -s	Make symbolic links instead of hard links

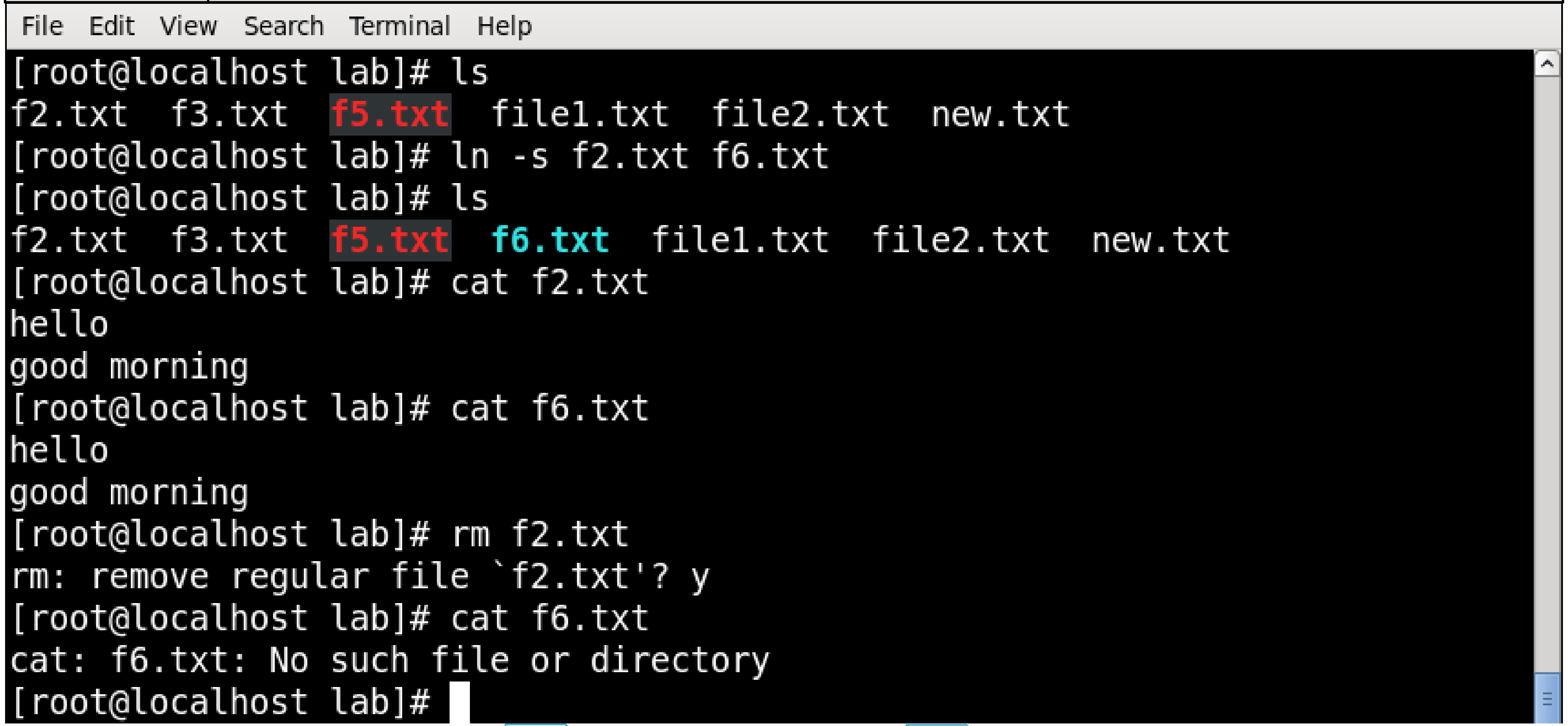
ln Command Example



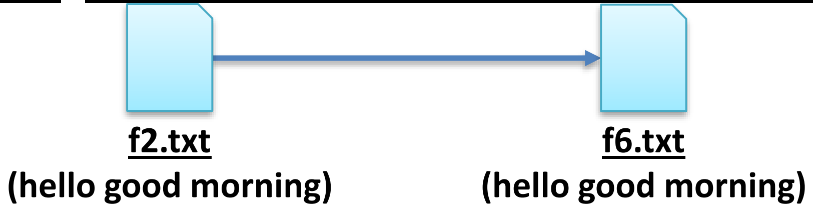
```
[root@localhost lab]# cat > f1.txt
hello linux
^C
[root@localhost lab]# link f1.txt new.txt
[root@localhost lab]# cat f1.txt
hello linux
[root@localhost lab]# cat new.txt
hello linux
[root@localhost lab]# echo "good morning" >> f1.txt
[root@localhost lab]# cat f1.txt
hello linux
good morning
[root@localhost lab]# cat new.txt
hello linux
good morning
[root@localhost lab]# rm f1.txt
rm: remove regular file `f1.txt'? y
[root@localhost lab]# cat new.txt
hello linux
good morning
[root@localhost lab]#
```

ln Command Example

ln -s	Make symbolic links instead of hard links



```
[root@localhost lab]# ls
f2.txt  f3.txt  f5.txt  file1.txt  file2.txt  new.txt
[root@localhost lab]# ln -s f2.txt f6.txt
[root@localhost lab]# ls
f2.txt  f3.txt  f5.txt  f6.txt  file1.txt  file2.txt  new.txt
[root@localhost lab]# cat f2.txt
hello
good morning
[root@localhost lab]# cat f6.txt
hello
good morning
[root@localhost lab]# rm f2.txt
rm: remove regular file `f2.txt'? y
[root@localhost lab]# cat f6.txt
cat: f6.txt: No such file or directory
[root@localhost lab]#
```



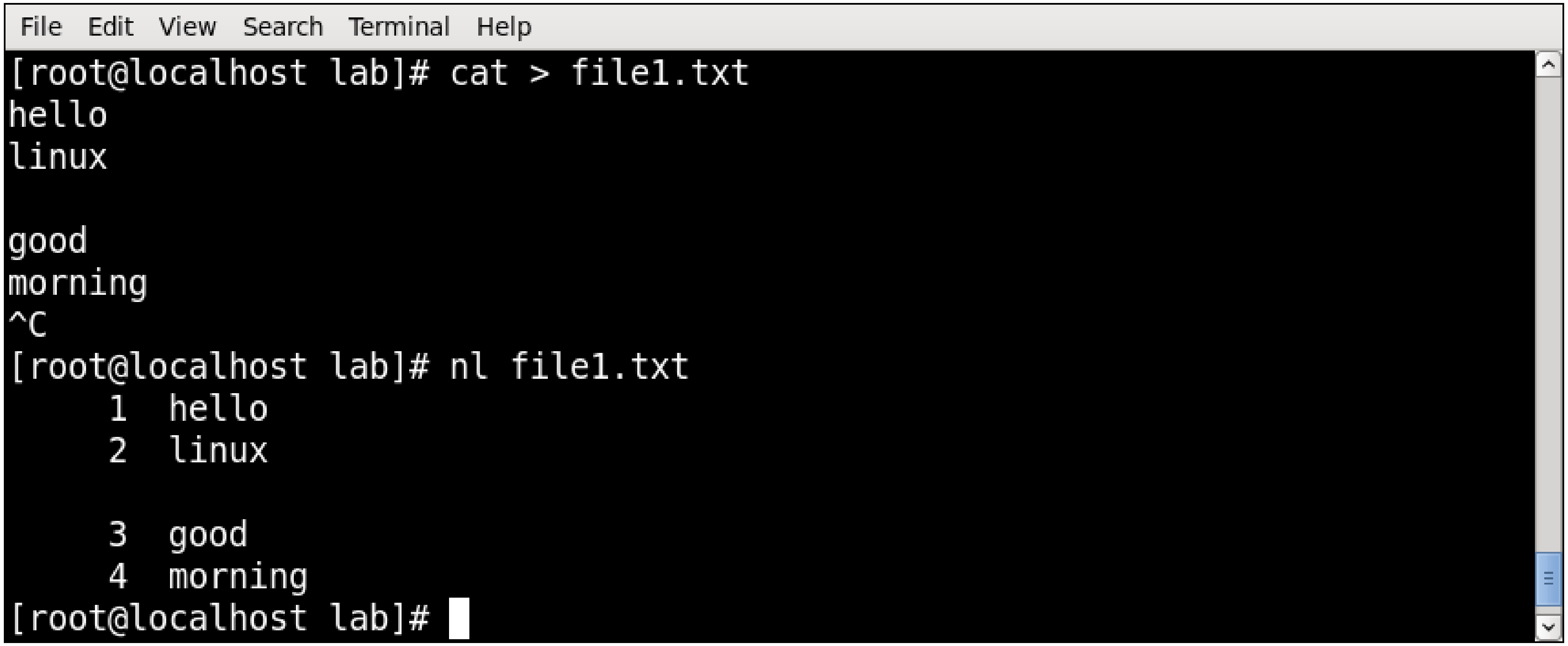
nl Command

- **nl** command numbers the lines in a file.
- **Syntax :**

  nl [OPTION]... [FILE]...
- **Example :**

Option	Use
nl -i	Line number increment at each line
nl -s	Add STRING after (possible) line number
nl -w	Use NUMBER columns for line numbers

nl Command Example



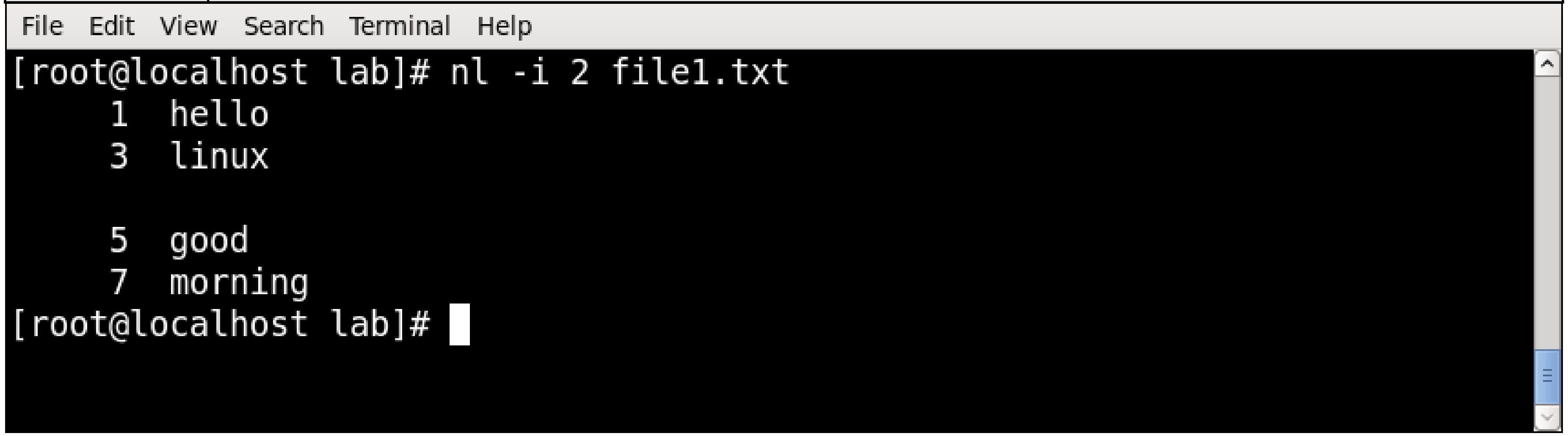
```
[root@localhost lab]# cat > file1.txt
hello
linux

good
morning
^C
[root@localhost lab]# nl file1.txt
     1  hello
     2  linux

     3  good
     4  morning
[root@localhost lab]# 
```

nl Command Example

nl -i	**Line number increment at each line**

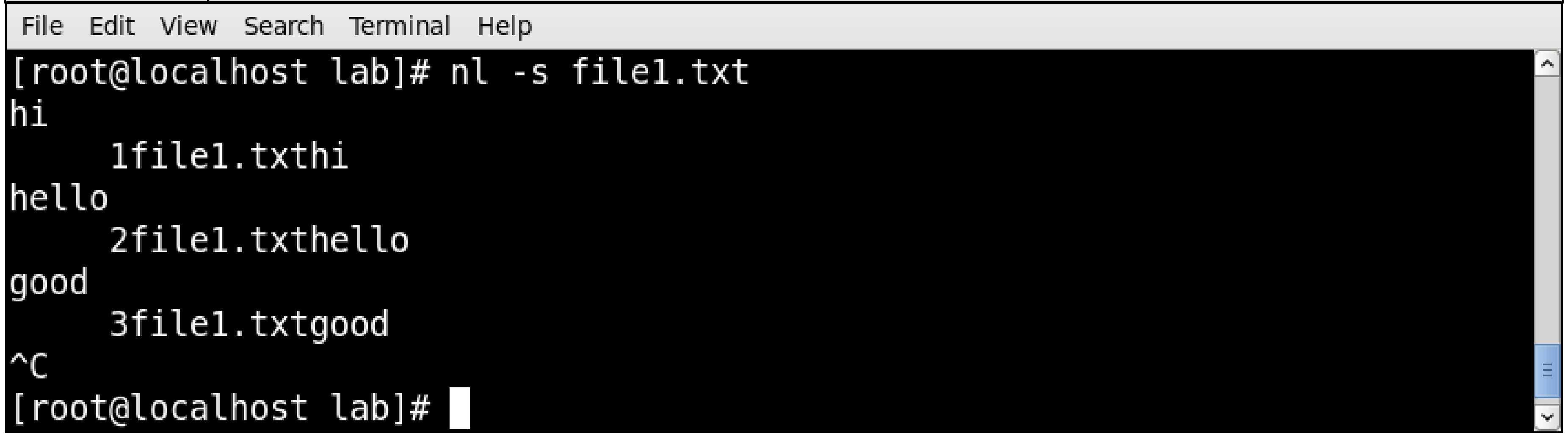


```
[root@localhost lab]# nl -i 2 file1.txt
     1  hello
     3  linux

     5  good
     7  morning
[root@localhost lab]# 
```

nl Command Example

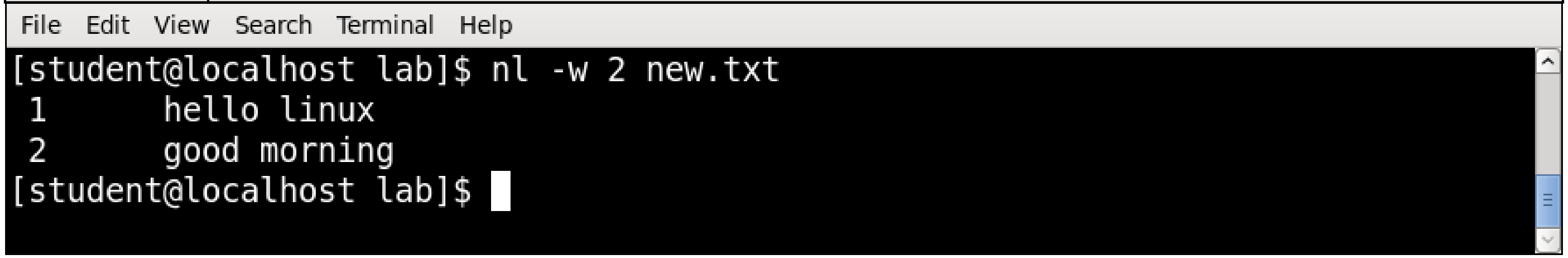
nl -s	**Add STRING after (possible) line number**



```
[root@localhost lab]# nl -s file1.txt
hi
     1file1.txthi
hello
     2file1.txthello
good
     3file1.txtgood
^C
[root@localhost lab]# 
```

nl Command Example

nl -w	**Use NUMBER columns for line numbers**



```
[student@localhost lab]$ nl -w 2 new.txt
 1      hello linux
 2      good morning
[student@localhost lab]$ 
```

head Command

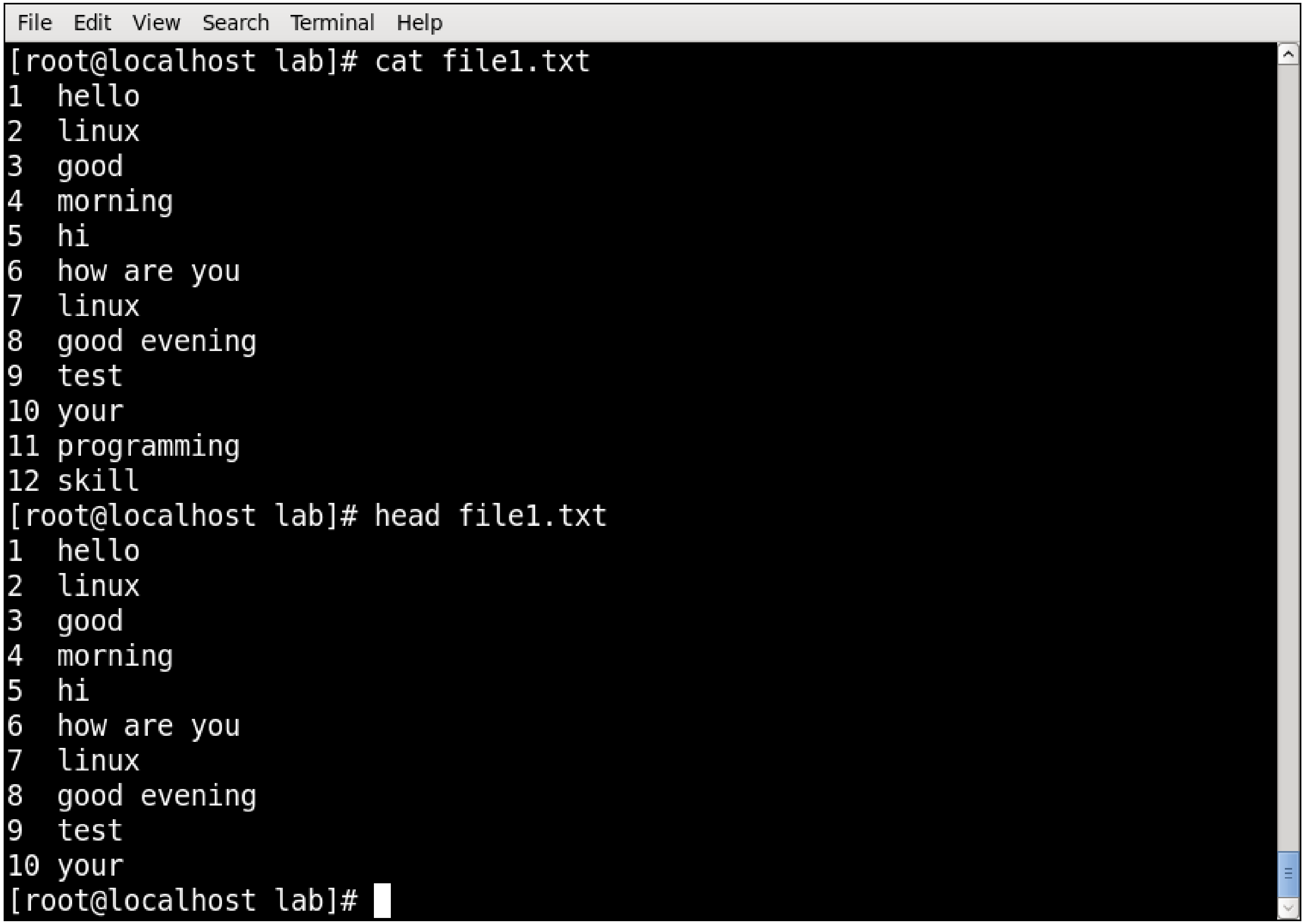
- **head** makes it easy to output the first part (10 lines by default) of files.
- **Syntax :**

  head [OPTION]... [FILE]...

- **Example :**

Option	Use
head -n	Print the first **n lines** instead of the first 10; with the leading '-', print all but the last **n** lines of each file
head -c	Print the first **n bytes** of each file; with a leading '-', print all but the last **n** bytes of each file
head -q	Never print headers identifying file names

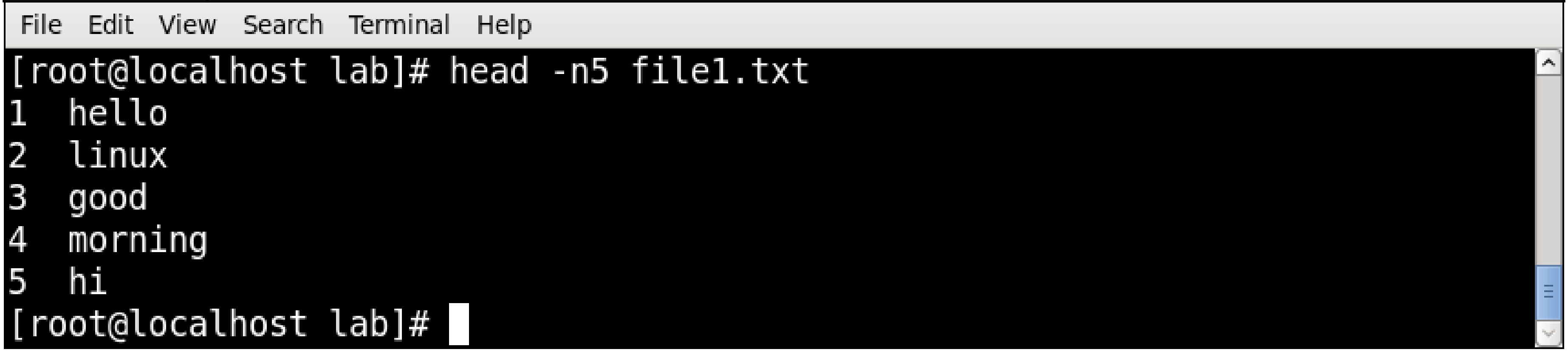
head Command Example



```
[root@localhost lab]# cat file1.txt
1  hello
2  linux
3  good
4  morning
5  hi
6  how are you
7  linux
8  good evening
9  test
10 your
11 programming
12 skill
[root@localhost lab]# head file1.txt
1  hello
2  linux
3  good
4  morning
5  hi
6  how are you
7  linux
8  good evening
9  test
10 your
[root@localhost lab]#
```

head Command Example

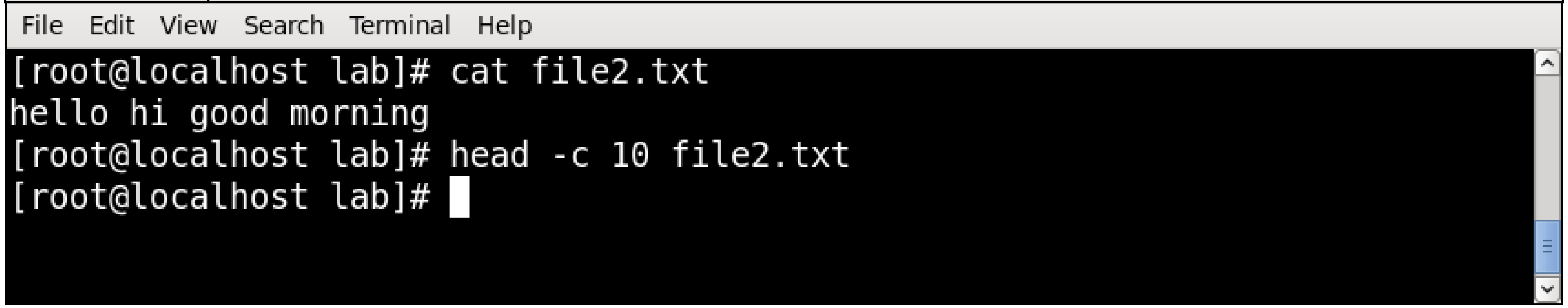
head -n	**Print the first n lines instead of the first 10; with the leading '-', print all but the last n lines of each file**



```
[root@localhost lab]# head -n5 file1.txt
1  hello
2  linux
3  good
4  morning
5  hi
[root@localhost lab]# 
```

head Command Example

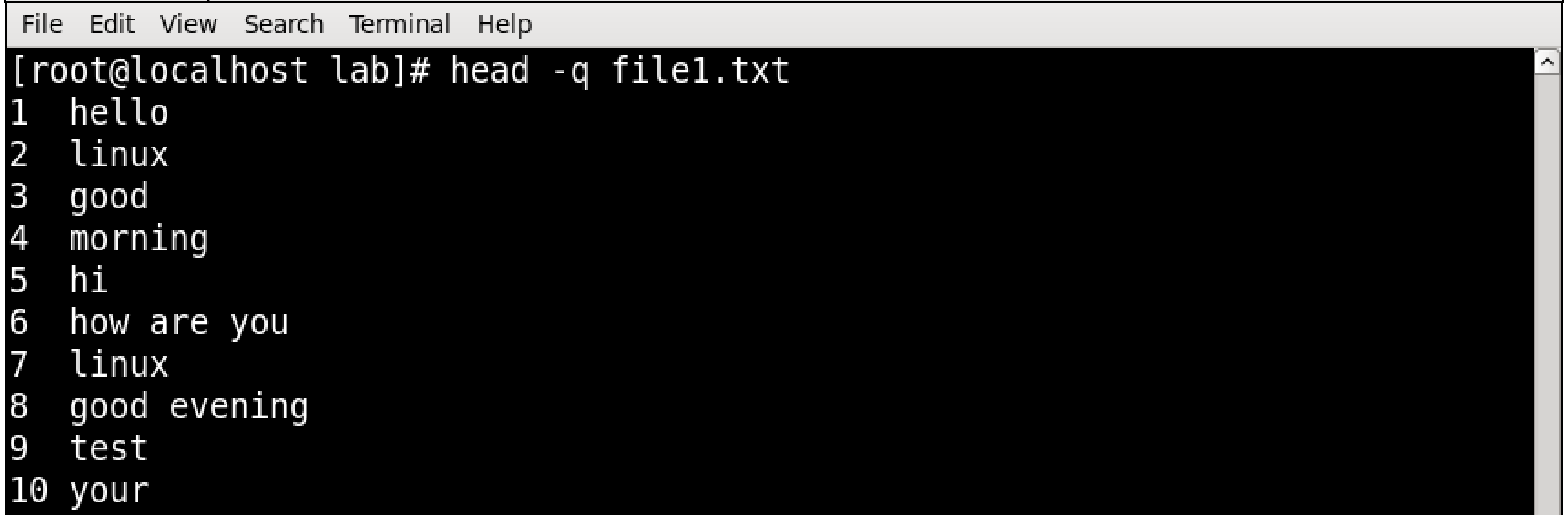
head -c	**Print the first n bytes of each file; with a leading '-', print all but the last n bytes of each file**



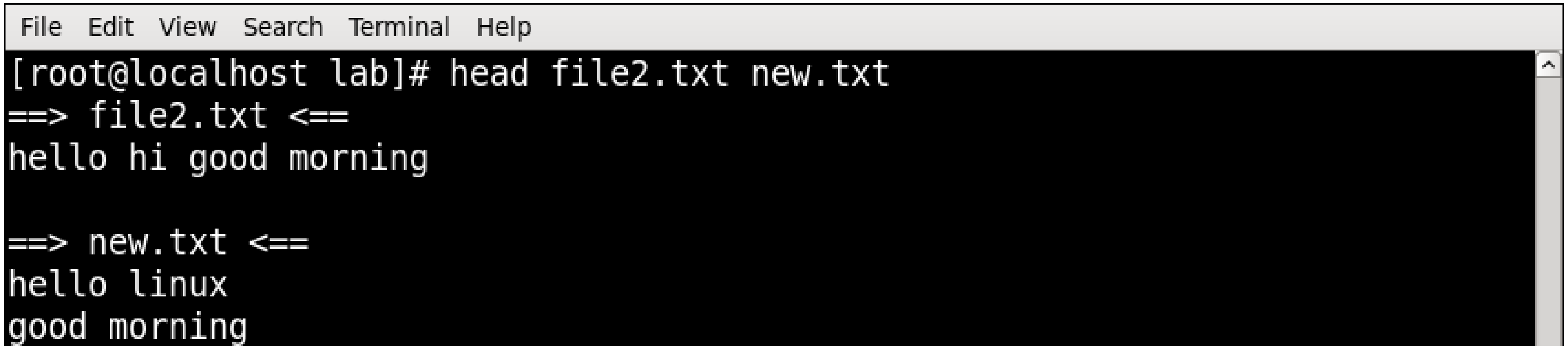
```
[root@localhost lab]# cat file2.txt
hello hi good morning
[root@localhost lab]# head -c 10 file2.txt
[root@localhost lab]# 
```

head Command Example

head -q	**Never print headers identifying file names**



```
[root@localhost lab]# head -q file1.txt
1  hello
2  linux
3  good
4  morning
5  hi
6  how are you
7  linux
8  good evening
9  test
10 your
```



```
[root@localhost lab]# head file2.txt new.txt
==> file2.txt <==
hello hi good morning

==> new.txt <==
hello linux
good morning
```

tail Command

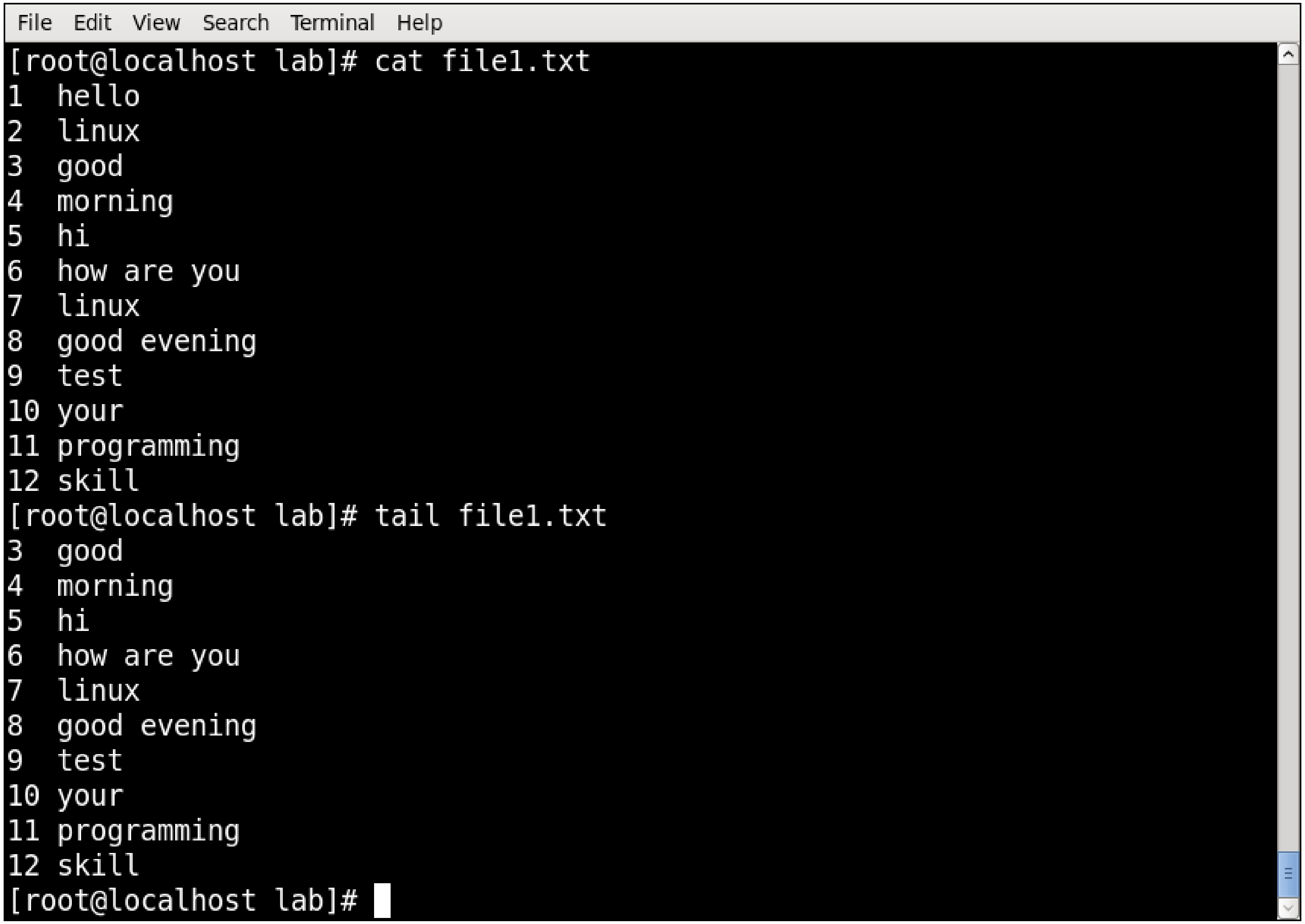
- **tail** is a command which prints the last few number of lines (10 lines by default) of a certain file, then terminates.

- **Syntax :**

```
tail [OPTION]... [FILE]...
```

Option	Use
tail -n	Output the last num lines, instead of the default (10)
tail -c	Output the last num bytes of each file
tail -q	Never output headers

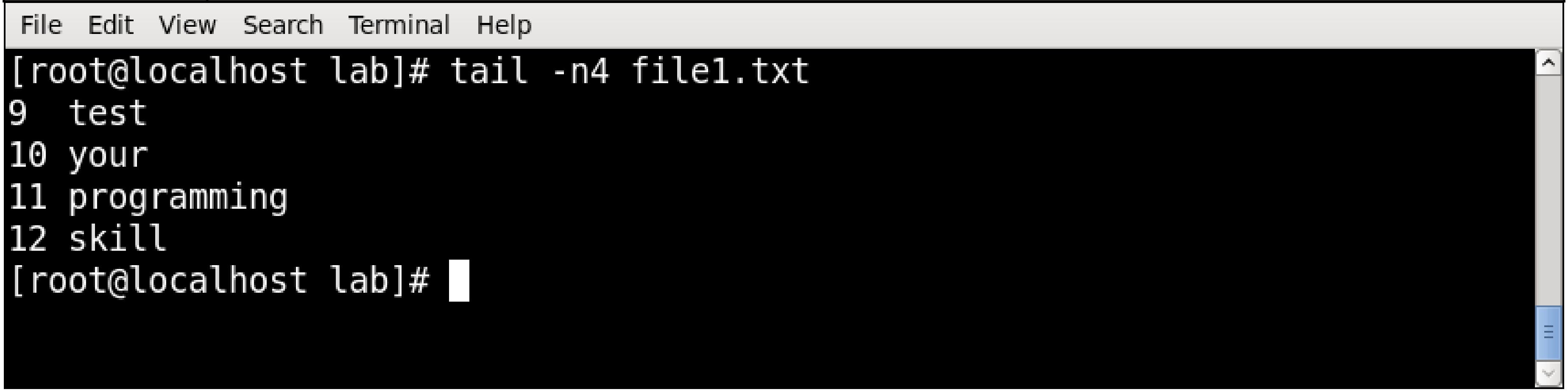
tail Command Example



```
[root@localhost lab]# cat file1.txt
1  hello
2  linux
3  good
4  morning
5  hi
6  how are you
7  linux
8  good evening
9  test
10 your
11 programming
12 skill
[root@localhost lab]# tail file1.txt
3  good
4  morning
5  hi
6  how are you
7  linux
8  good evening
9  test
10 your
11 programming
12 skill
[root@localhost lab]#
```

tail Command Example

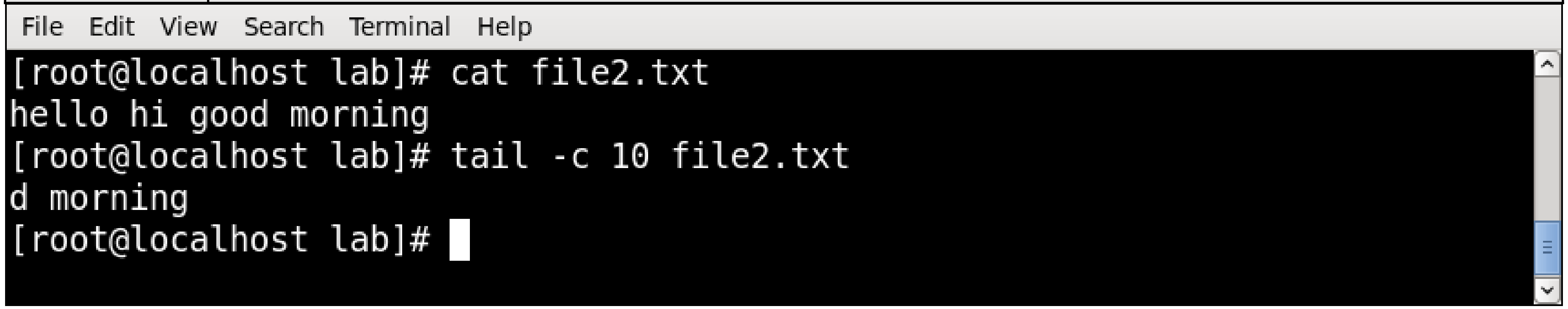
tail -n	**Output the last num lines, instead of the default (10)**



```
[root@localhost lab]# tail -n4 file1.txt
9  test
10 your
11 programming
12 skill
[root@localhost lab]# 
```

tail Command Example

tail -c	**Output the last num bytes of each file**



```
[root@localhost lab]# cat file2.txt
hello hi good morning
[root@localhost lab]# tail -c 10 file2.txt
d morning
[root@localhost lab]#
```

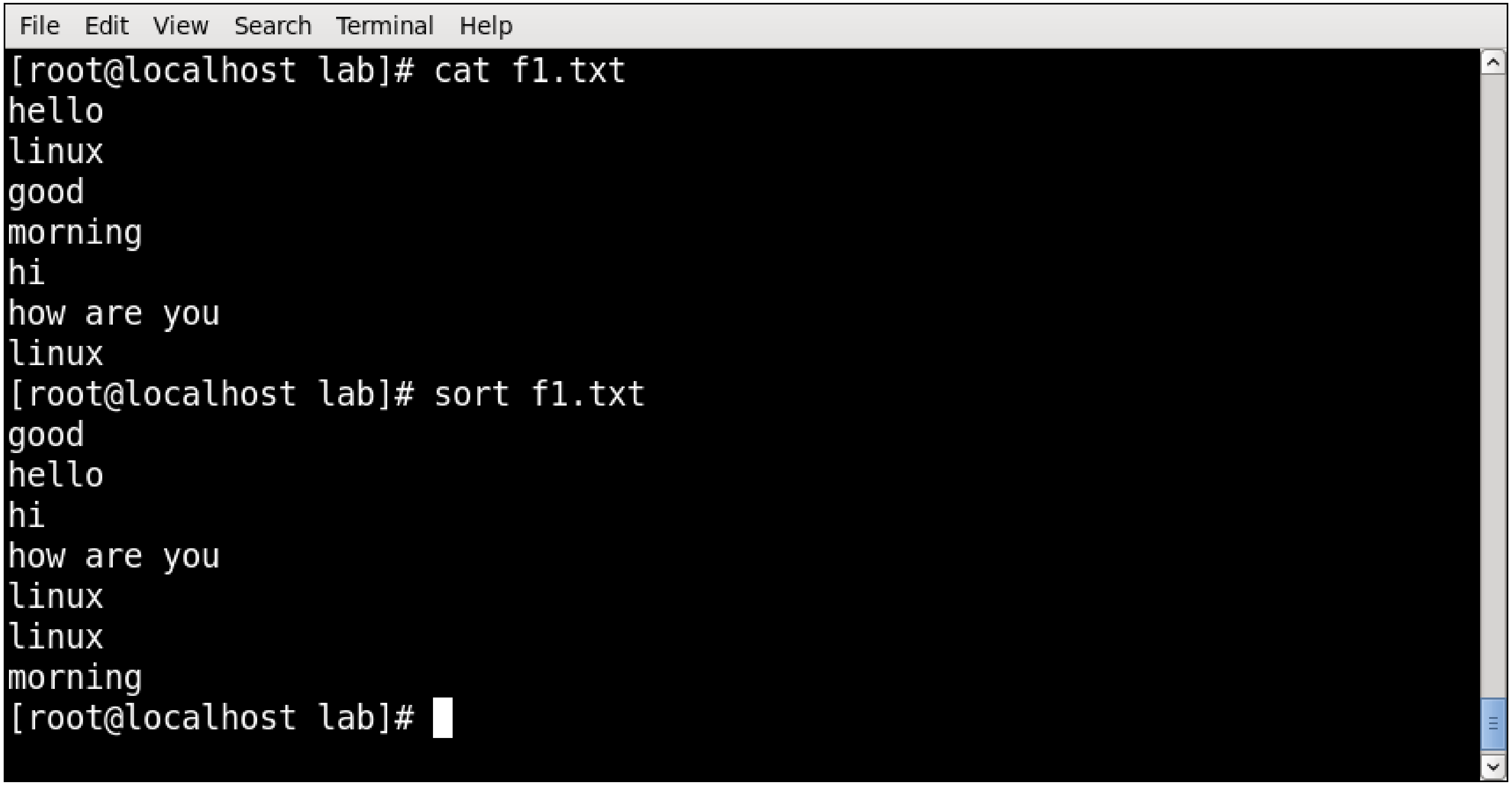
sort Command

- **sort** command is used to sort a file, arranging the records in a particular order.
- By default, the sort command sorts file assuming the contents are ASCII. Using options in sort command, it can also be used to sort numerically.

- **Syntax :** sort [OPTION]... [FILE]...

Option	Use
sort -c	To check if the file given is already sorted or not
sort -r	Reverse the result of comparisons
sort -n	Compare according to string numerical value
sort -nr	To sort a file with numeric data in reverse order
sort -k	Sorting a table on the basis of any column
sort -b	Ignore leading blanks

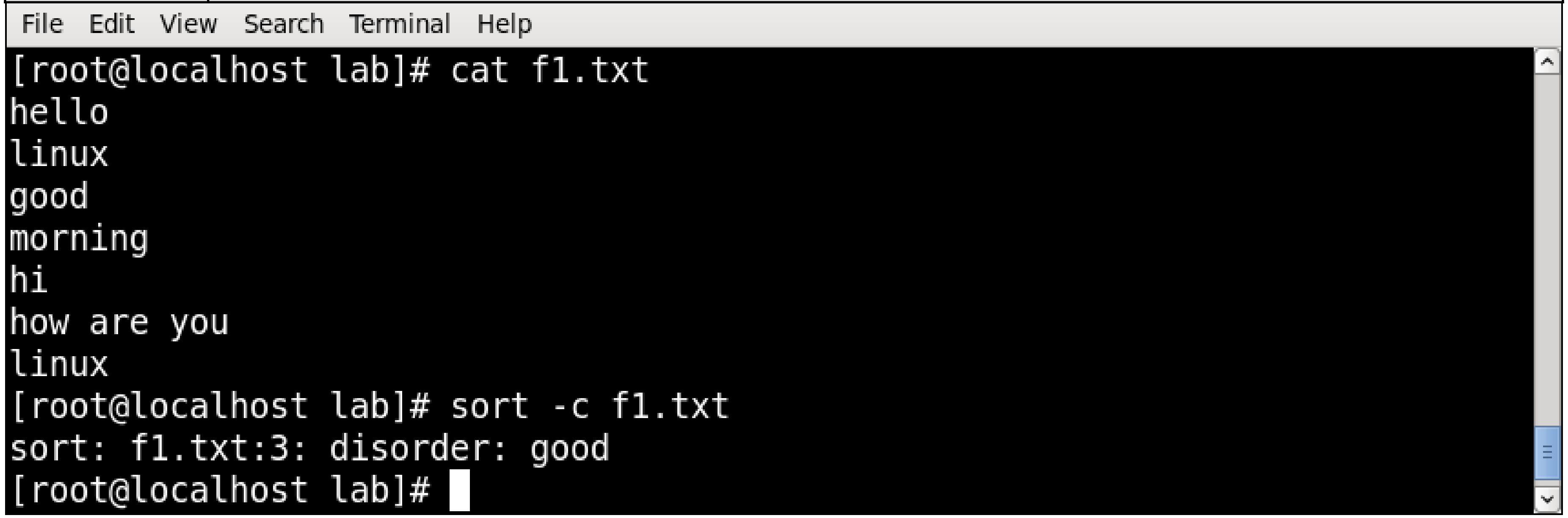
sort Command Example



```
[root@localhost lab]# cat f1.txt
hello
linux
good
morning
hi
how are you
linux
[root@localhost lab]# sort f1.txt
good
hello
hi
how are you
linux
linux
morning
[root@localhost lab]#
```

sort Command Example

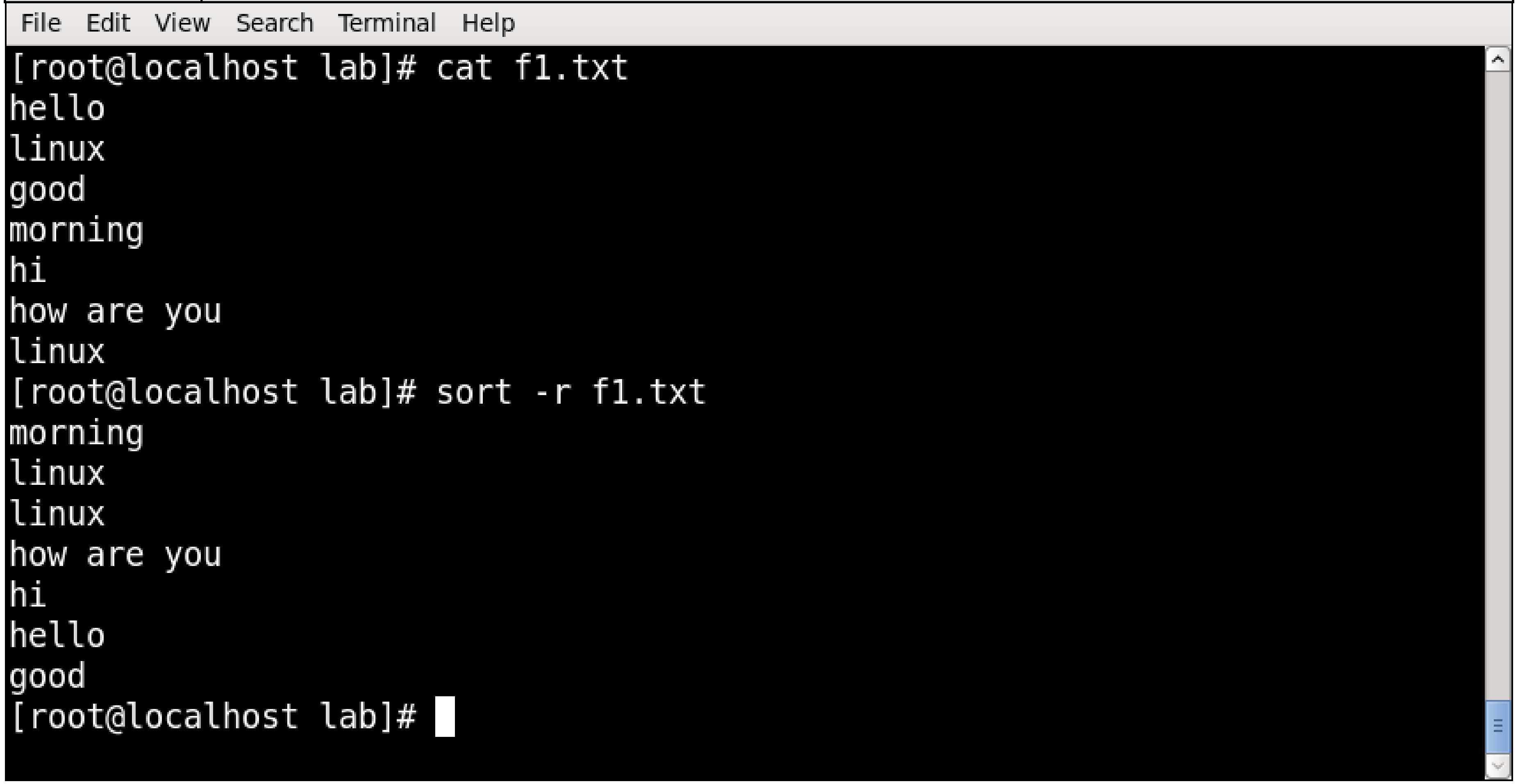
sort -c	**To check if the file given is already sorted or not**


File Edit View Search Terminal Help

```
[root@localhost lab]# cat f1.txt
hello
linux
good
morning
hi
how are you
linux
[root@localhost lab]# sort -c f1.txt
sort: f1.txt:3: disorder: good
[root@localhost lab]# 
```

sort Command Example

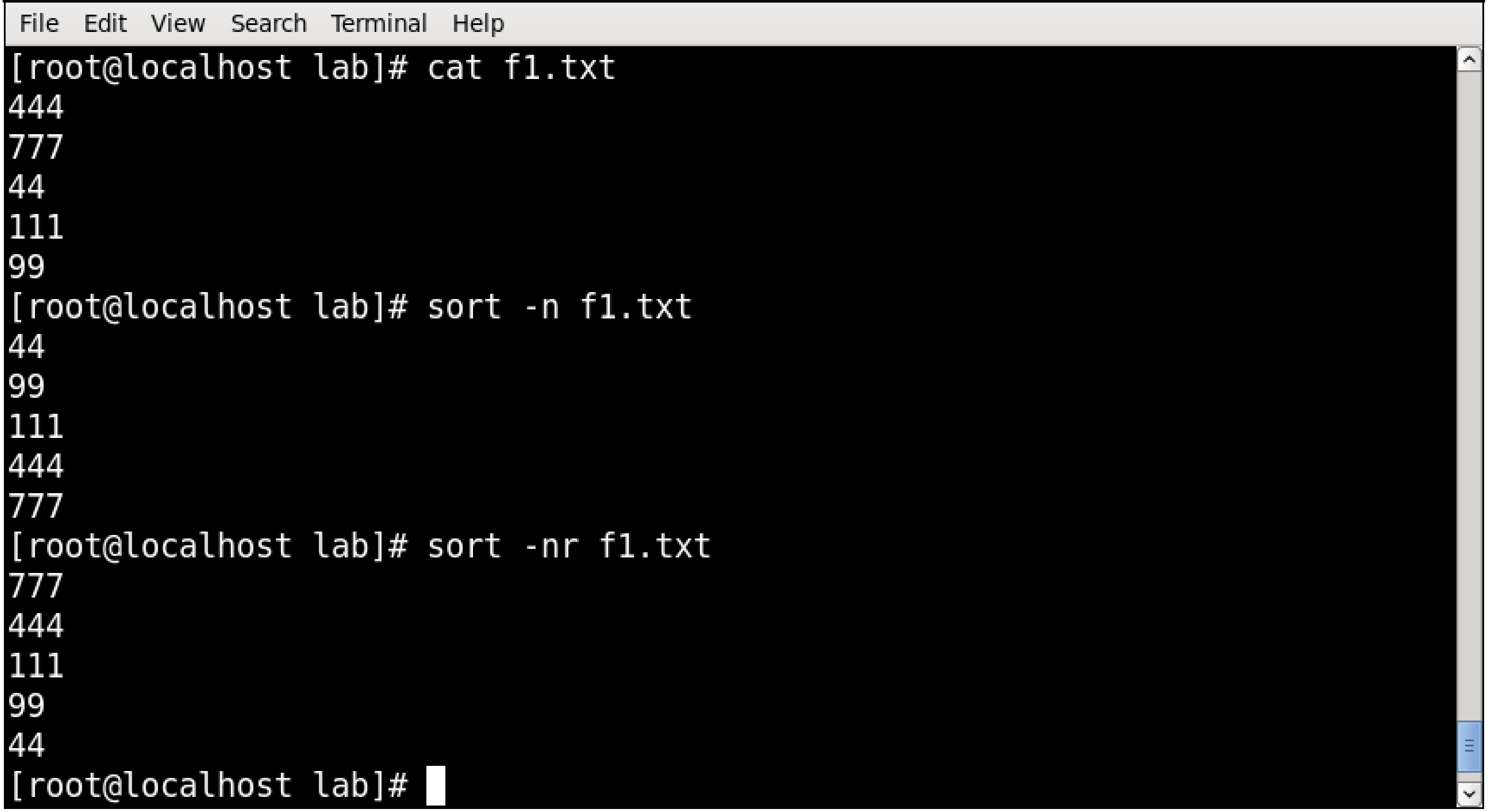
sort -r	**Reverse the result of comparisons**



```
[root@localhost lab]# cat f1.txt
hello
linux
good
morning
hi
how are you
linux
[root@localhost lab]# sort -r f1.txt
morning
linux
linux
how are you
hi
hello
good
[root@localhost lab]#
```

sort Command Example

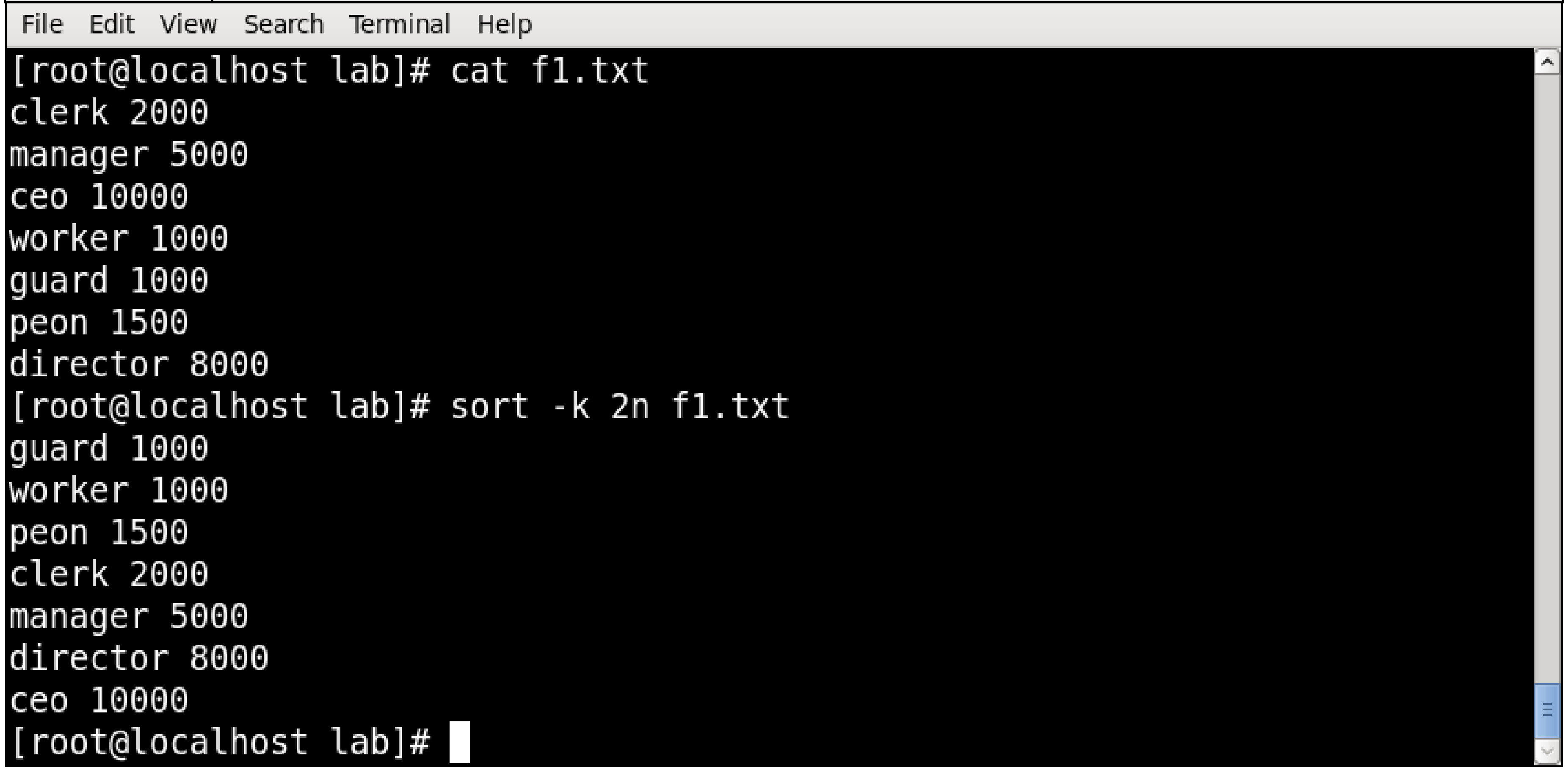
sort -n	**Compare according to string numerical value**
sort -nr	**To sort a file with numeric data in reverse order**



```
[root@localhost lab]# cat f1.txt
444
777
44
111
99
[root@localhost lab]# sort -n f1.txt
44
99
111
444
777
[root@localhost lab]# sort -nr f1.txt
777
444
111
99
44
[root@localhost lab]# 
```

sort Command Example

sort -k	**Sorting a table on the basis of any column**



```
[root@localhost lab]# cat f1.txt
clerk 2000
manager 5000
ceo 10000
worker 1000
guard 1000
peon 1500
director 8000
[root@localhost lab]# sort -k 2n f1.txt
guard 1000
worker 1000
peon 1500
clerk 2000
manager 5000
director 8000
ceo 10000
[root@localhost lab]# 
```

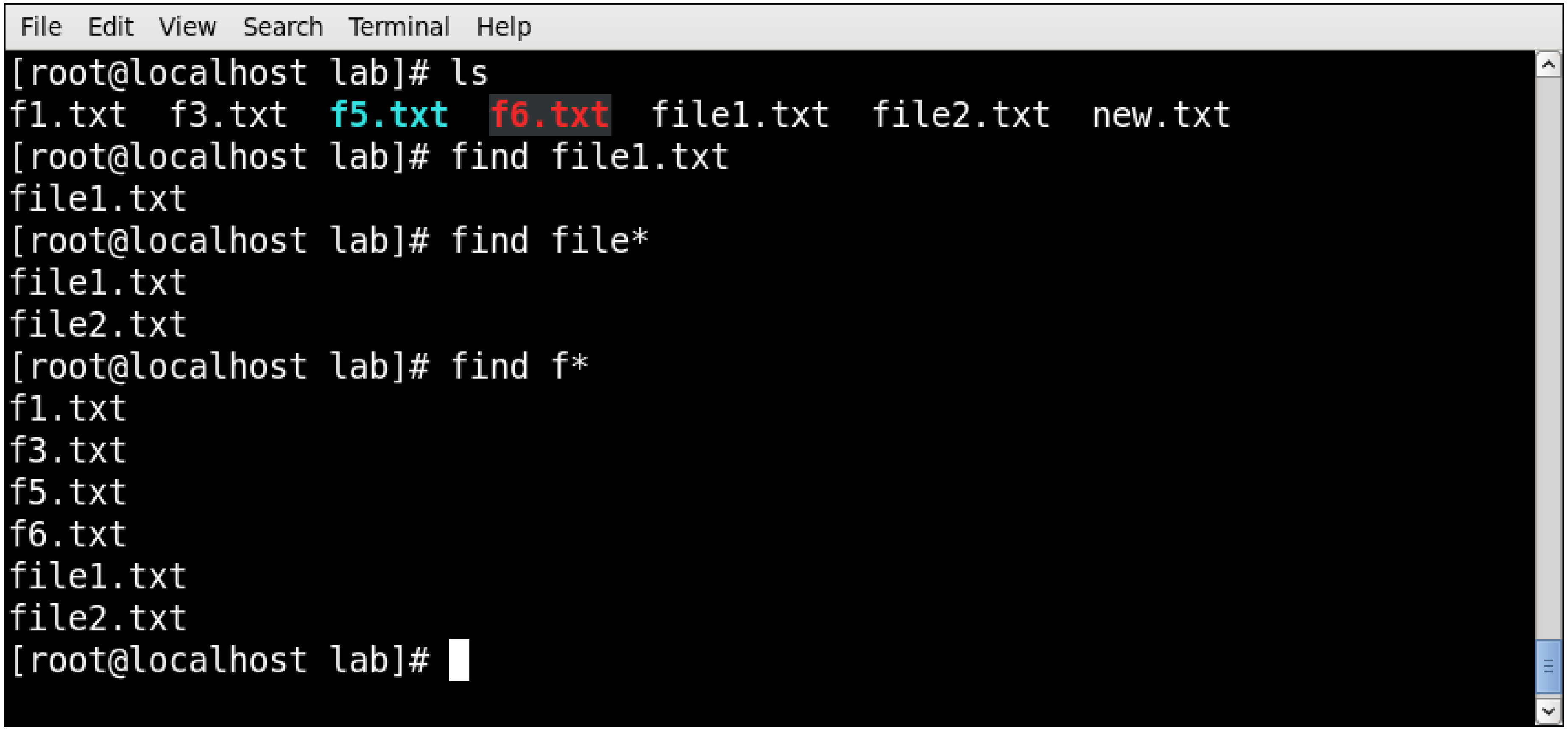
find Command

- **find** command searches for files in a directory hierarchy.
- **Syntax :**

  find [option] [path...] [expression]

Option	Use
find -name filename	Search for files that are specified by ‘filename’
find -newer filename	Search for files that were modified/created after ‘filename’
find -user name	Search for files owned by user name or ID ‘name’
find -size +N/-N	Search for files of ‘N’ blocks; ‘N’ followed by ‘c’ can be used to measure size in characters
find -empty	Search for empty files and directories
find -perm octal	Search for the file if permission is ‘octal’

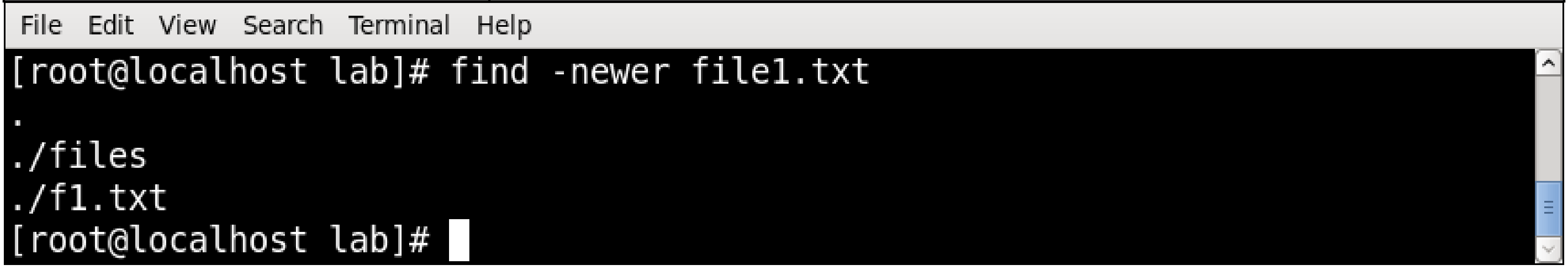
find Command Example



```
[root@localhost lab]# ls
f1.txt  f3.txt  f5.txt  f6.txt  file1.txt  file2.txt  new.txt
[root@localhost lab]# find file1.txt
file1.txt
[root@localhost lab]# find file*
file1.txt
file2.txt
[root@localhost lab]# find f*
f1.txt
f3.txt
f5.txt
f6.txt
file1.txt
file2.txt
[root@localhost lab]# 
```

find Command Example

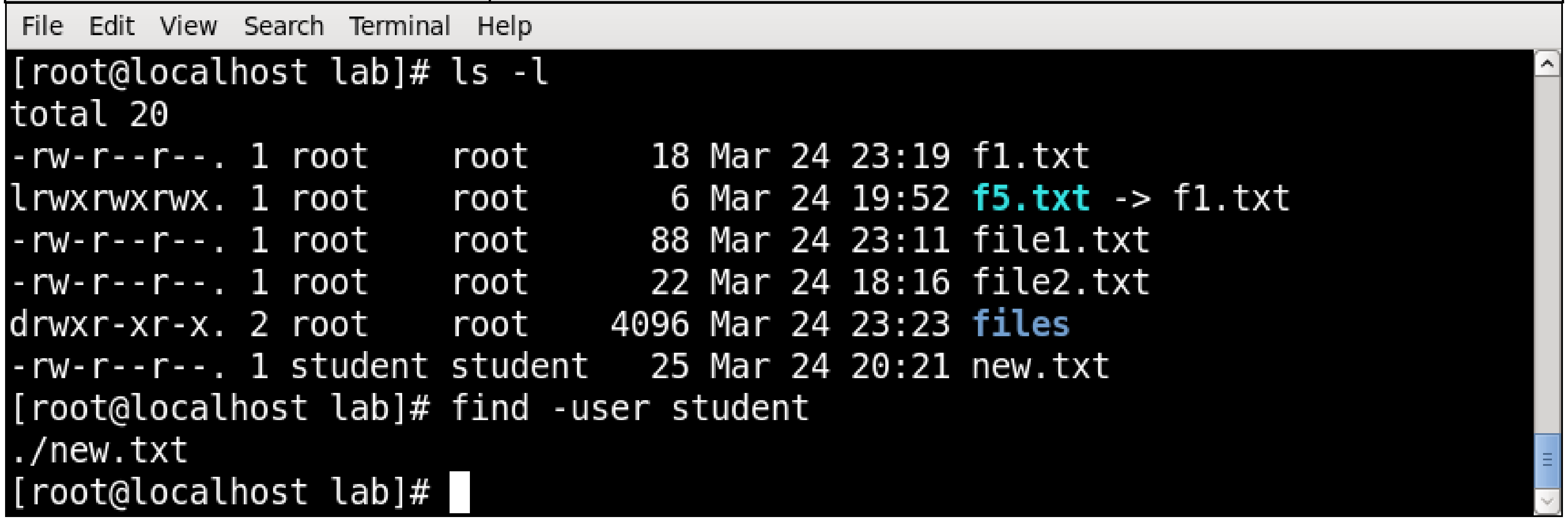
find -newer filename	**Search for files that were modified/created after 'filename'**



```
[root@localhost lab]# find -newer file1.txt
.
./files
./f1.txt
[root@localhost lab]# 
```

find Command Example

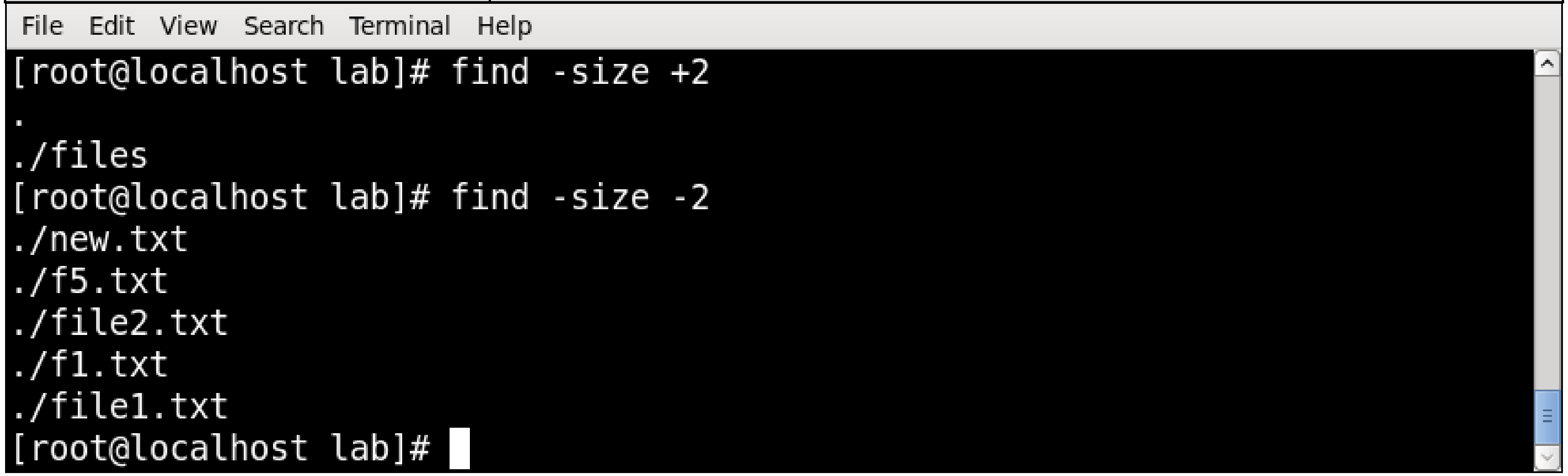
find -user name	**Search for files owned by user name or ID ‘name’**



```
[root@localhost lab]# ls -l
total 20
-rw-r--r--. 1 root    root      18 Mar 24 23:19 f1.txt
lrwxrwxrwx. 1 root    root       6 Mar 24 19:52 f5.txt -> f1.txt
-rw-r--r--. 1 root    root      88 Mar 24 23:11 file1.txt
-rw-r--r--. 1 root    root      22 Mar 24 18:16 file2.txt
drwxr-xr-x. 2 root    root    4096 Mar 24 23:23 files
-rw-r--r--. 1 student student   25 Mar 24 20:21 new.txt
[root@localhost lab]# find -user student
./new.txt
[root@localhost lab]#
```

find Command Example

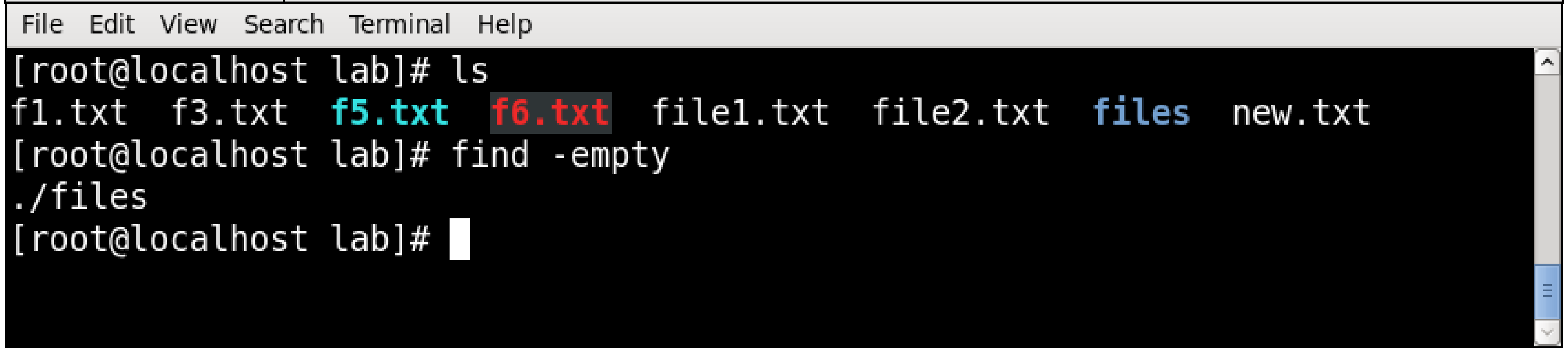
find -size +N/-N	**Search for files of ‘N’ blocks; ‘N’ followed by ‘c’ can be used to measure size in characters**



```
[root@localhost lab]# find -size +2
.
./files
[root@localhost lab]# find -size -2
./new.txt
./f5.txt
./file2.txt
./f1.txt
./file1.txt
[root@localhost lab]# 
```

find Command Example

find -empty	**Search for empty files and directories**



```
[root@localhost lab]# ls
f1.txt  f3.txt  f5.txt  f6.txt  file1.txt  file2.txt  files  new.txt
[root@localhost lab]# find -empty
./files
[root@localhost lab]# 
```

uniq Command

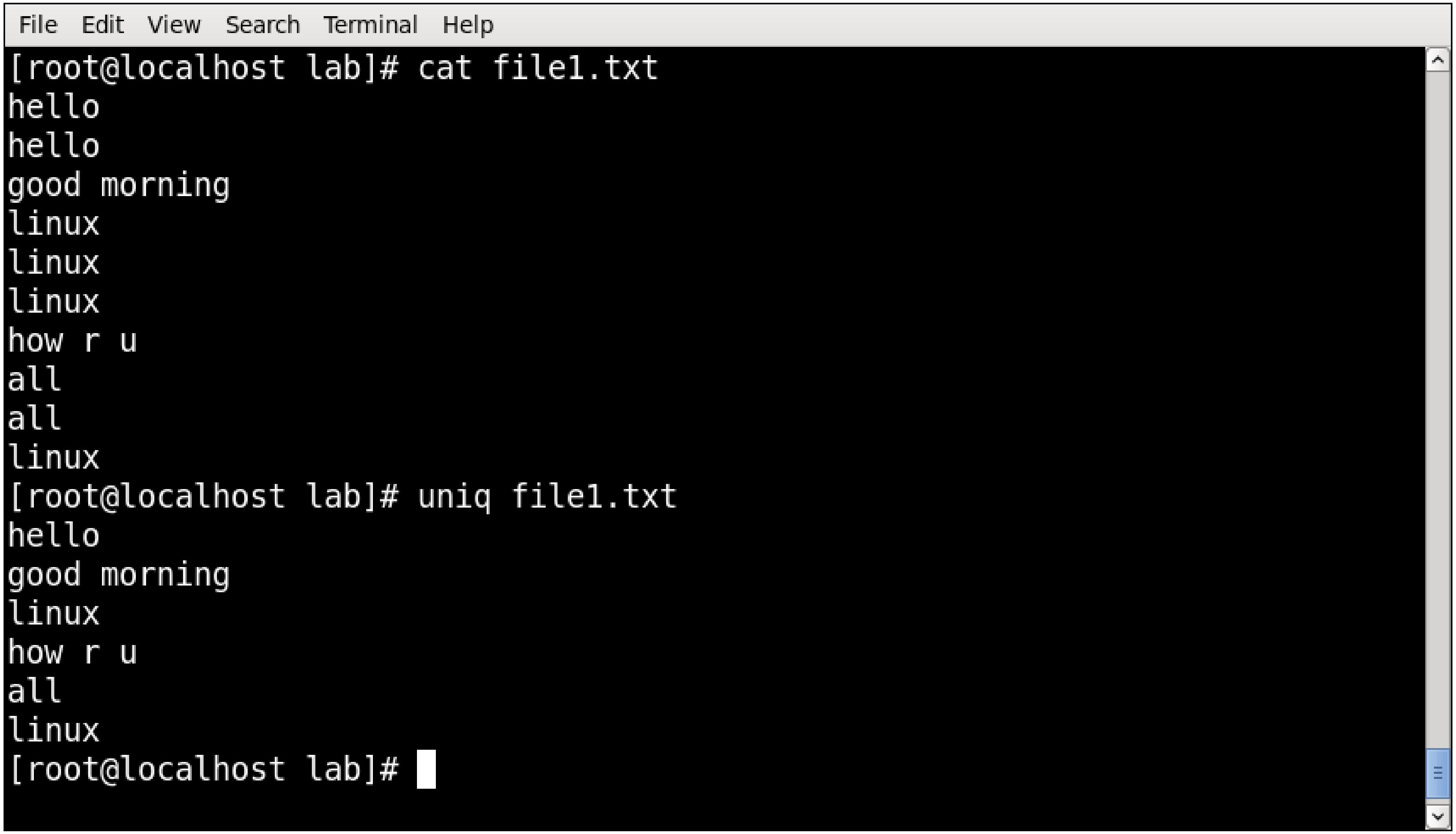
- **uniq** reports or filters out repeated lines in a file.
- It can remove duplicates, show a count of occurrences, show only repeated lines, ignore certain characters and compare on specific fields.

- **Syntax :**

  uniq [OPTION]... [INPUT [OUTPUT]]

Option	Use
uniq -u	Prints only unique lines
uniq -d	Only print duplicated lines
uniq -D	Print all duplicate lines
uniq -c	Prefix lines with a number representing how many times they occurred
uniq -i	Ignore case when comparing

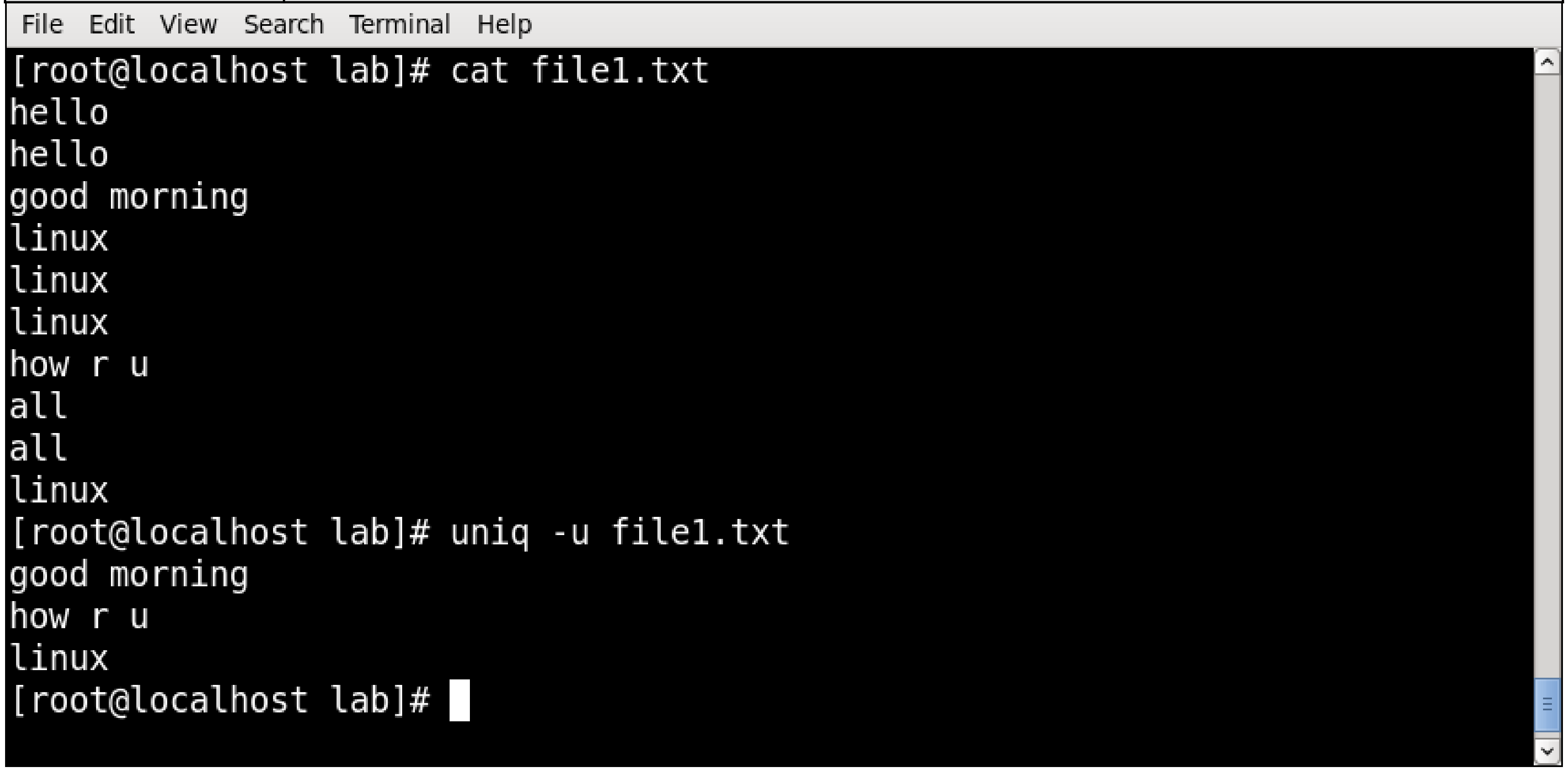
uniq Command Example



```
[root@localhost lab]# cat file1.txt
hello
hello
good morning
linux
linux
linux
how r u
all
all
linux
[root@localhost lab]# uniq file1.txt
hello
good morning
linux
how r u
all
linux
[root@localhost lab]# 
```

uniq Command Example

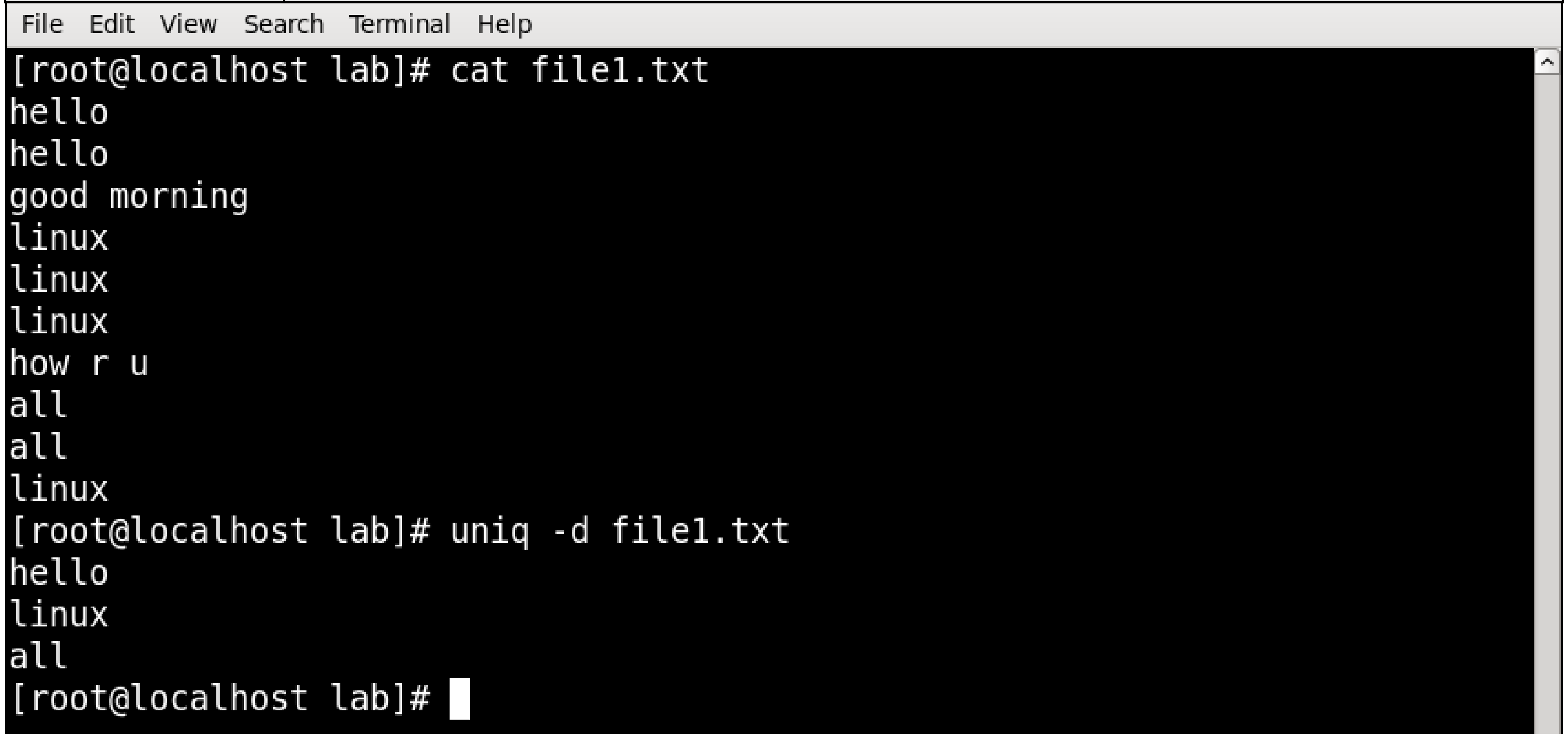
uniq -u	**Prints only unique lines**



```
[root@localhost lab]# cat file1.txt
hello
hello
good morning
linux
linux
linux
how r u
all
all
linux
[root@localhost lab]# uniq -u file1.txt
good morning
how r u
linux
[root@localhost lab]#
```

uniq Command Example

uniq -d	**Only print duplicated lines**



```
[root@localhost lab]# cat file1.txt
hello
hello
good morning
linux
linux
linux
how r u
all
all
linux
[root@localhost lab]# uniq -d file1.txt
hello
linux
all
[root@localhost lab]# 
```

grep Command

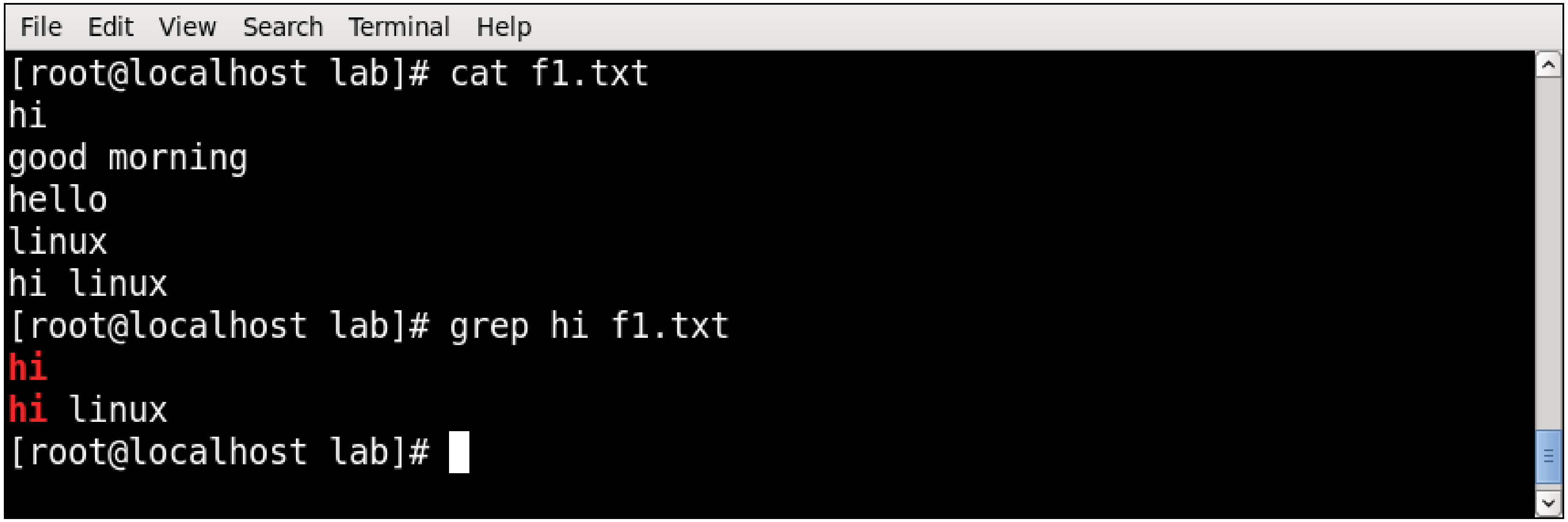
- The **grep** filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern.
- The pattern that is searched in the file is referred to as the regular expression.
- grep stands for globally search for regular expression and print out.

▪ **Syntax :**

```
grep [options] pattern [files]
```

Option	Use
grep -c	Prints only a count of the lines that match a pattern
grep -h	Display the matched lines, but do not display the filenames
grep -l	Displays list of a filenames only
grep -i	Ignores, case for matching

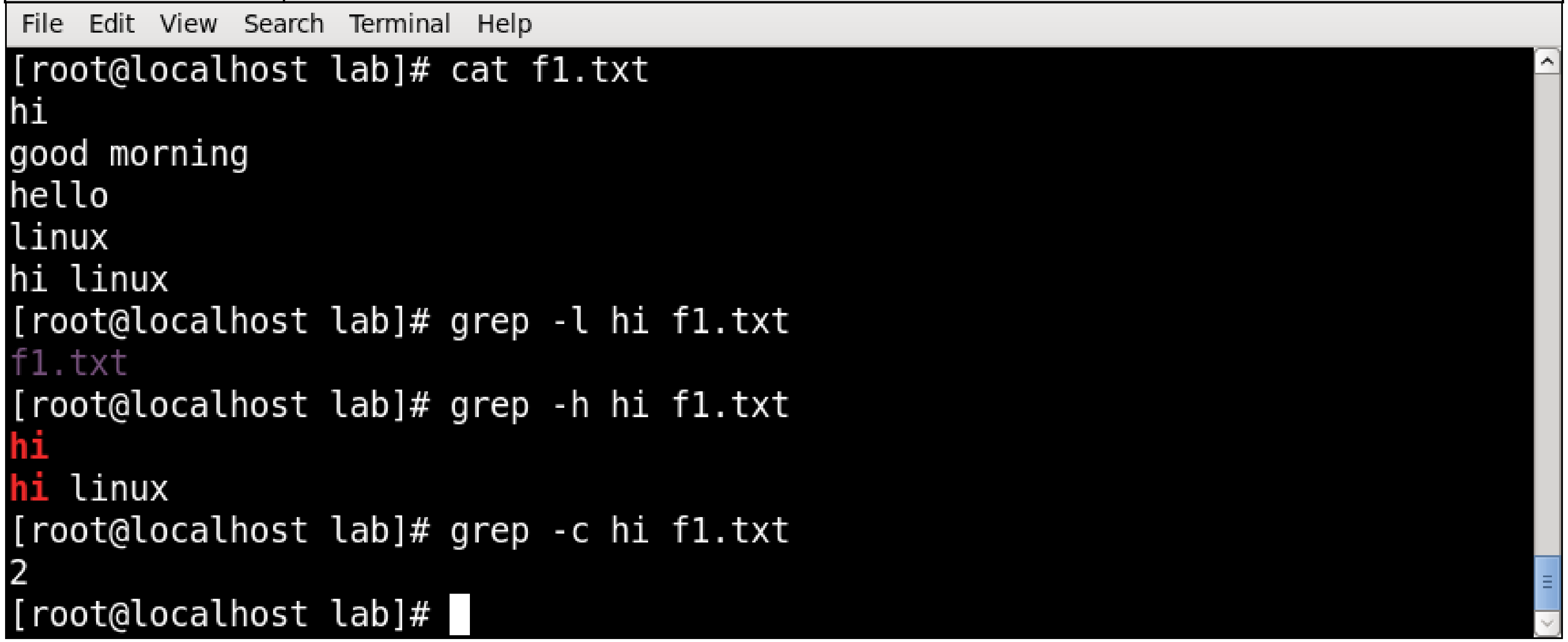
grep Command Example



```
[root@localhost lab]# cat f1.txt
hi
good morning
hello
linux
hi linux
[root@localhost lab]# grep hi f1.txt
hi
hi linux
[root@localhost lab]#
```

grep Command Example

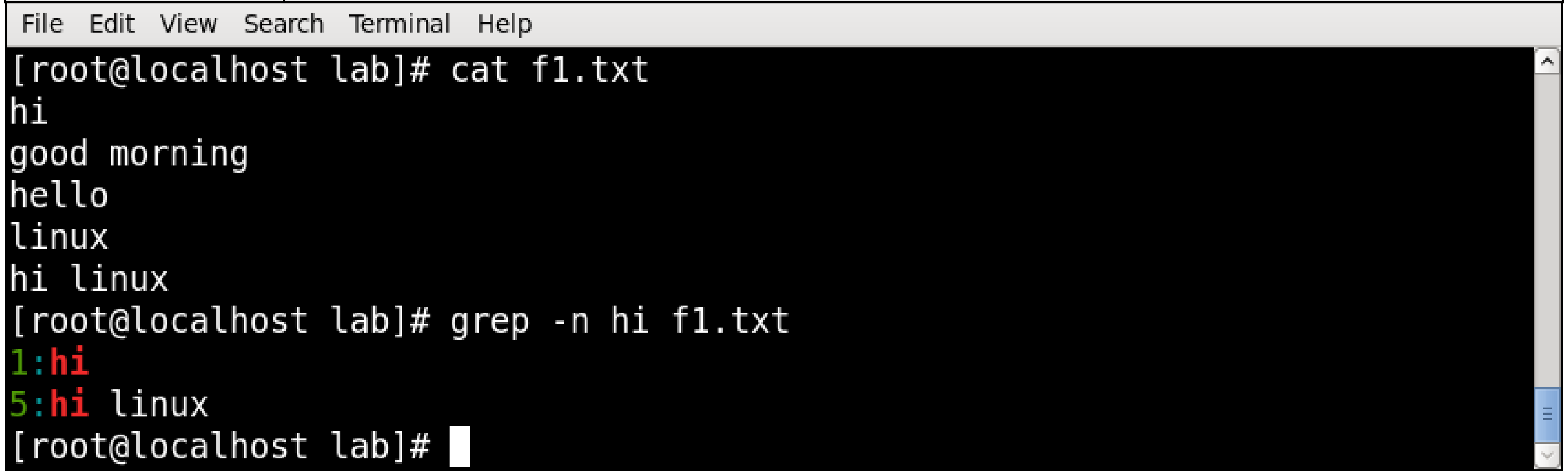
grep -c	**Prints only a count of the lines that match a pattern**
grep -h	**Display the matched lines, but do not display the filenames**
grep -l	**Displays list of a filenames only**



```
[root@localhost lab]# cat f1.txt
hi
good morning
hello
linux
hi linux
[root@localhost lab]# grep -l hi f1.txt
f1.txt
[root@localhost lab]# grep -h hi f1.txt
hi
hi linux
[root@localhost lab]# grep -c hi f1.txt
2
[root@localhost lab]# 
```

grep Command Example

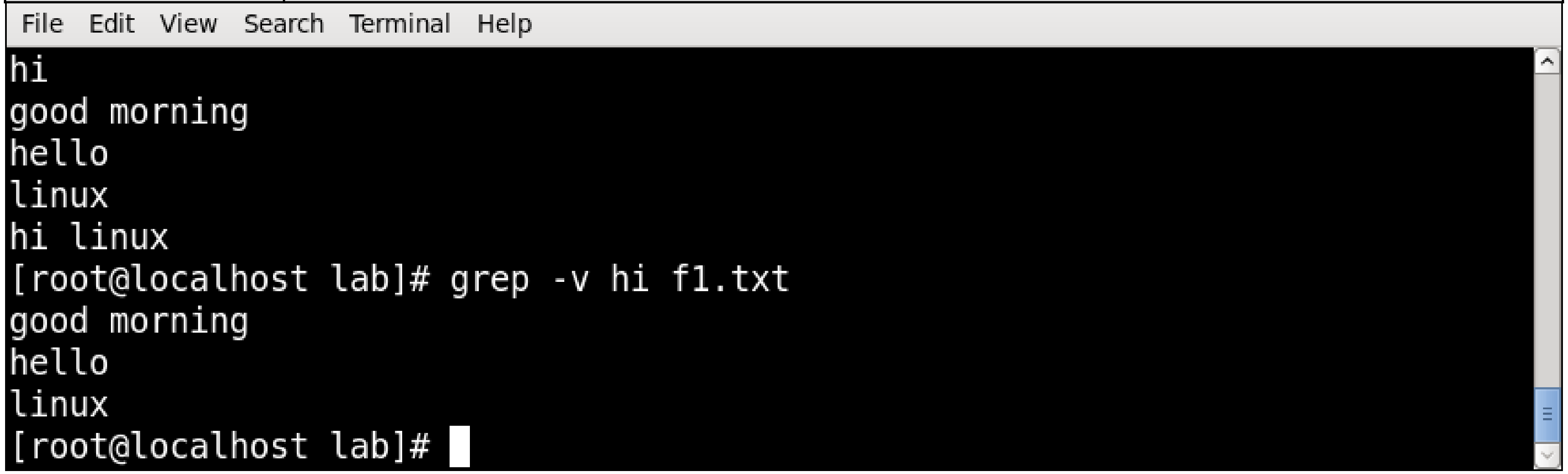
grep -n	**Display the matched lines and their line numbers**



```
[root@localhost lab]# cat f1.txt
hi
good morning
hello
linux
hi linux
[root@localhost lab]# grep -n hi f1.txt
1:hi
5:hi linux
[root@localhost lab]#
```

grep Command Example

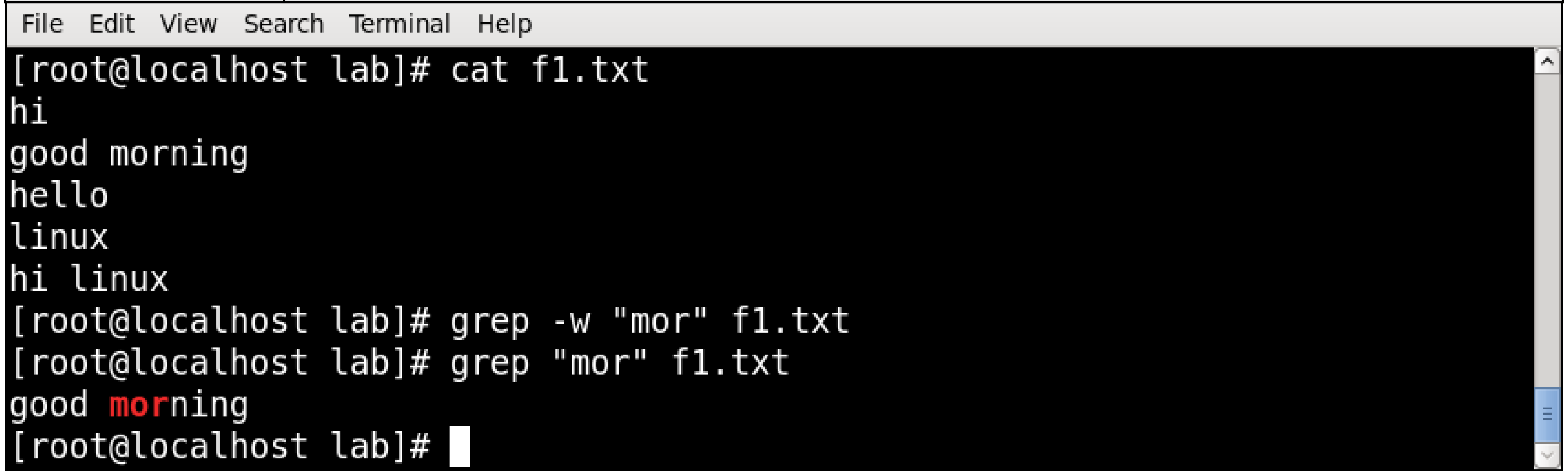
grep -v	**This prints out all the lines that do not matches the pattern**



```
hi
good morning
hello
linux
hi linux
[root@localhost lab]# grep -v hi f1.txt
good morning
hello
linux
[root@localhost lab]#
```

grep Command Example

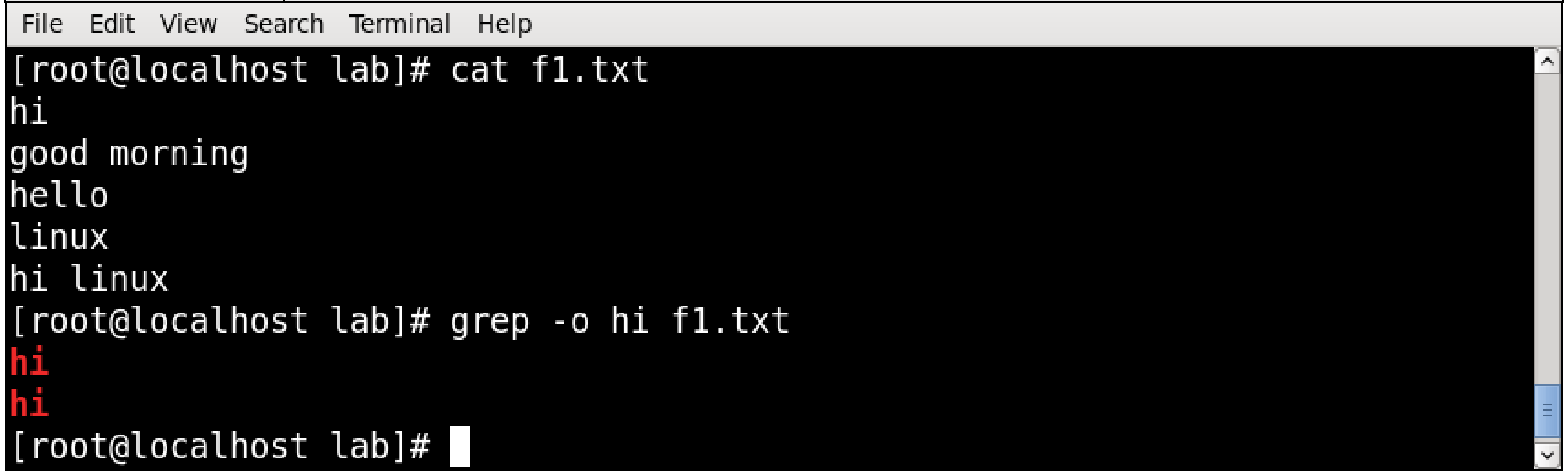
grep -w	**Match whole word**



```
[root@localhost lab]# cat f1.txt
hi
good morning
hello
linux
hi linux
[root@localhost lab]# grep -w "mor" f1.txt
[root@localhost lab]# grep "mor" f1.txt
good morning
[root@localhost lab]#
```

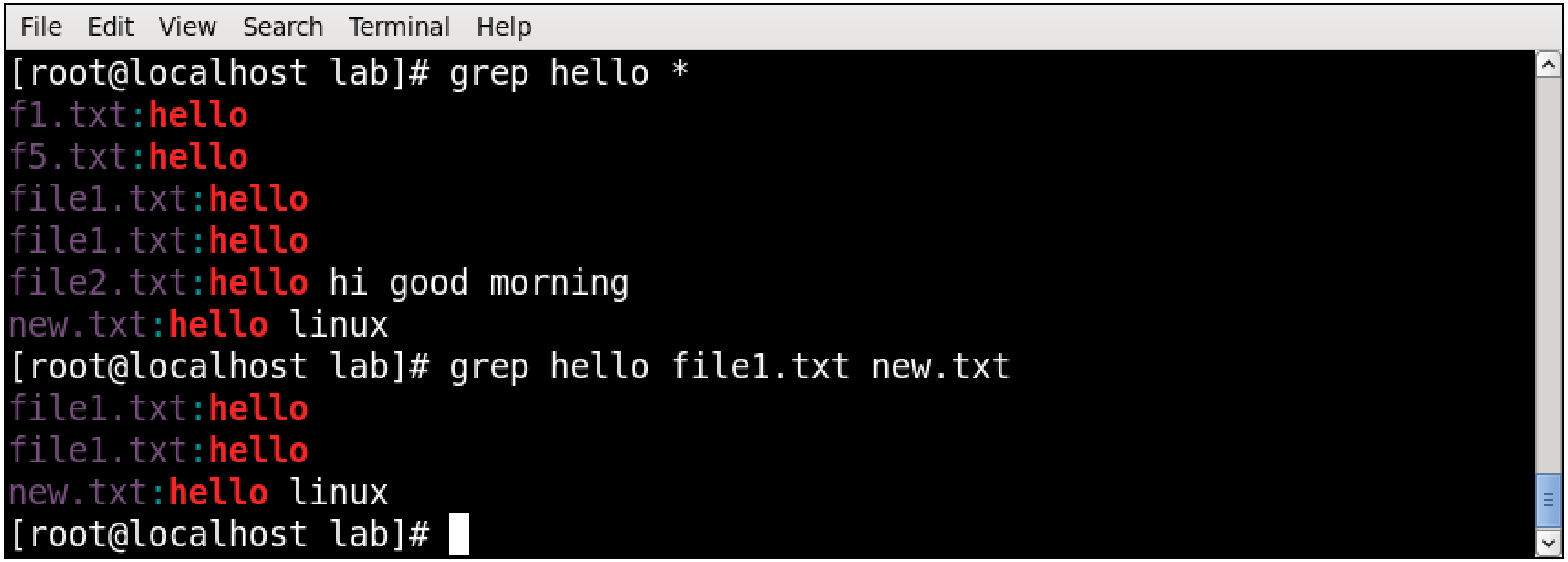
grep Command Example

grep -o	**Print only the matched parts of a matching line**



```
[root@localhost lab]# cat f1.txt
hi
good morning
hello
linux
hi linux
[root@localhost lab]# grep -o hi f1.txt
hi
hi
[root@localhost lab]#
```

grep Command Example



```
[root@localhost lab]# grep hello *
f1.txt:hello
f5.txt:hello
file1.txt:hello
file1.txt:hello
file2.txt:hello hi good morning
new.txt:hello linux
[root@localhost lab]# grep hello file1.txt new.txt
file1.txt:hello
file1.txt:hello
new.txt:hello linux
[root@localhost lab]# 
```

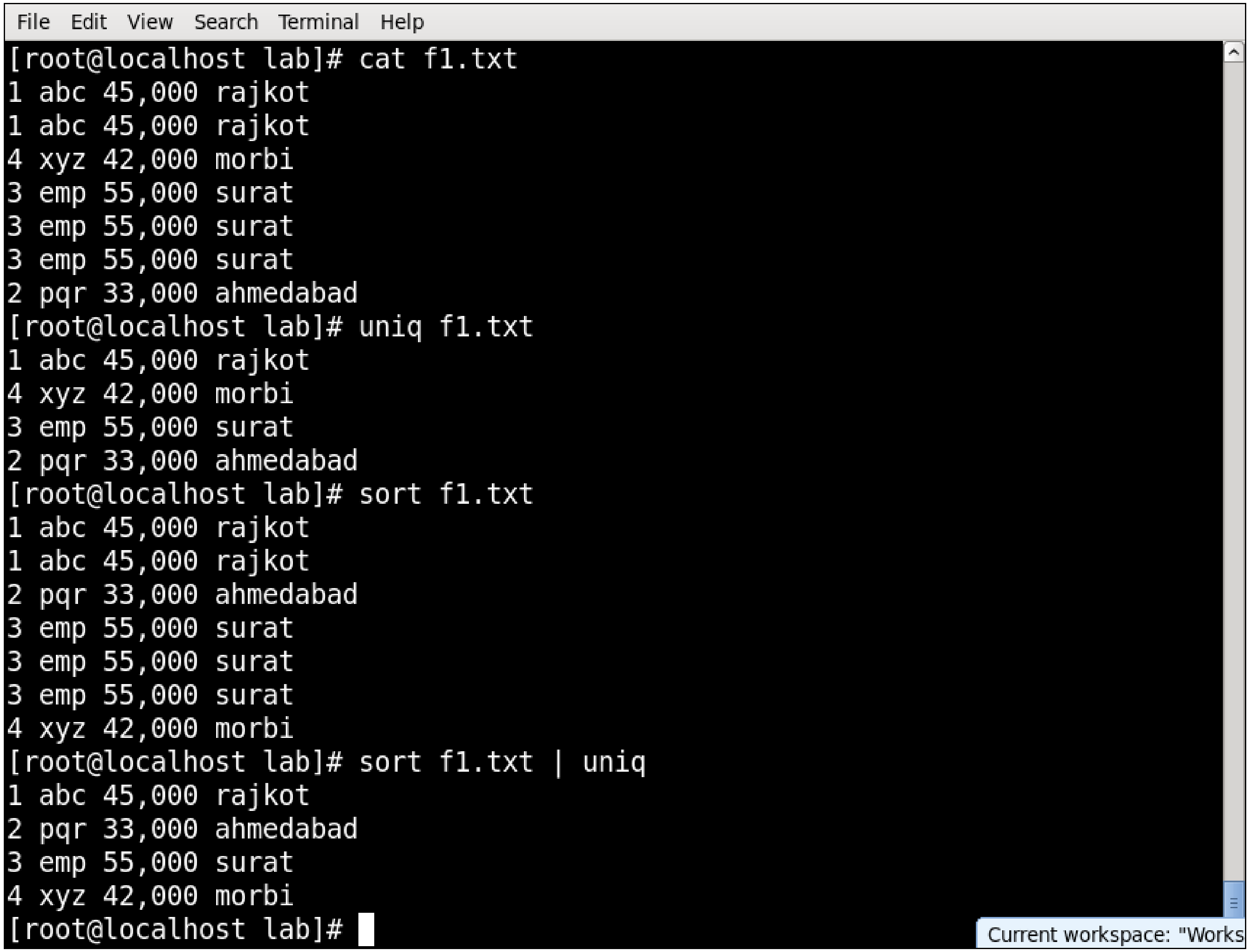
pipe (|) Command

- It redirects the command STDOUT or standard output into the given next command STDIN or standard input.
- In short, the output of each process directly as input to the next one like a pipeline.
- The symbol '**|**' denotes a **pipe**.
- Pipes help you mash-up two or more commands at the same time and run them consecutively.

- **Syntax :**

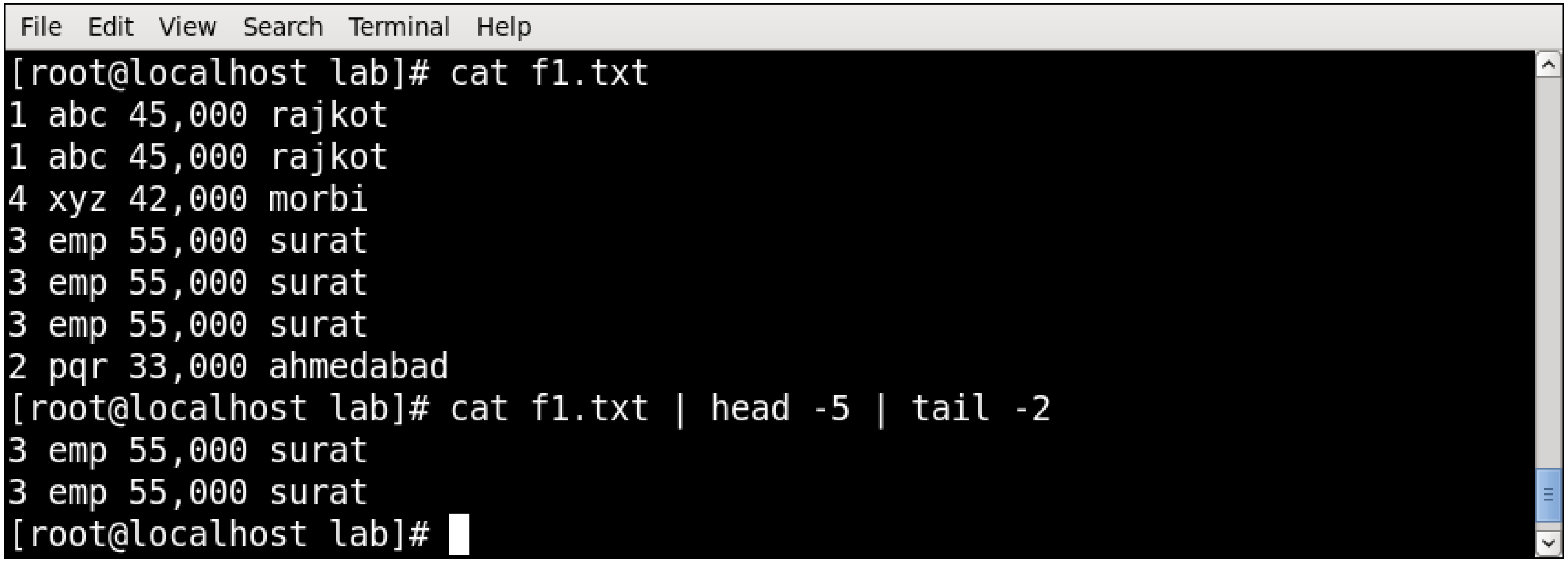
```
command_1 | command_2 | command_3 | .... | command_N...
```

pipe Command Example


File Edit View Search Terminal Help
Current workspace: "Works

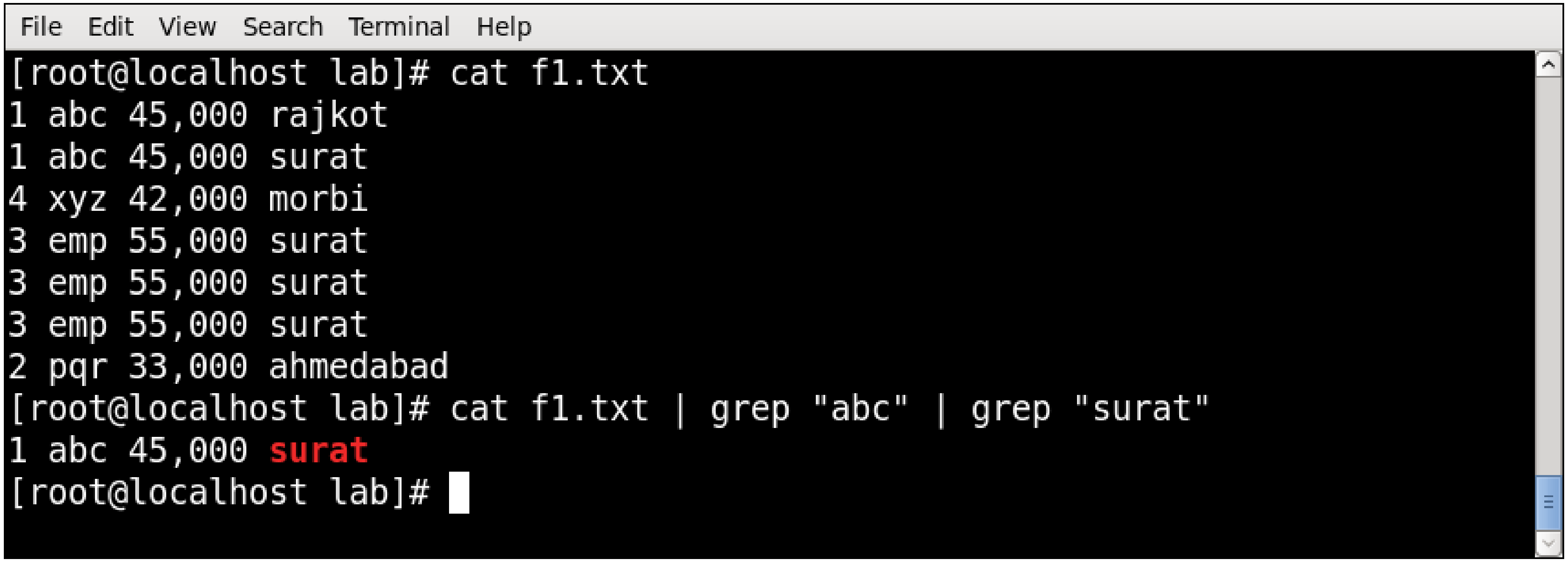
```
[root@localhost lab]# cat f1.txt
1 abc 45,000 rajkot
1 abc 45,000 rajkot
4 xyz 42,000 morbi
3 emp 55,000 surat
3 emp 55,000 surat
3 emp 55,000 surat
2 pqr 33,000 ahmedabad
[root@localhost lab]# uniq f1.txt
1 abc 45,000 rajkot
4 xyz 42,000 morbi
3 emp 55,000 surat
2 pqr 33,000 ahmedabad
[root@localhost lab]# sort f1.txt
1 abc 45,000 rajkot
1 abc 45,000 rajkot
2 pqr 33,000 ahmedabad
3 emp 55,000 surat
3 emp 55,000 surat
3 emp 55,000 surat
4 xyz 42,000 morbi
[root@localhost lab]# sort f1.txt | uniq
1 abc 45,000 rajkot
2 pqr 33,000 ahmedabad
3 emp 55,000 surat
4 xyz 42,000 morbi
[root@localhost lab]# 
```

pipe Command Example



```
[root@localhost lab]# cat f1.txt
1 abc 45,000 rajkot
1 abc 45,000 rajkot
4 xyz 42,000 morbi
3 emp 55,000 surat
3 emp 55,000 surat
3 emp 55,000 surat
2 pqr 33,000 ahmedabad
[root@localhost lab]# cat f1.txt | head -5 | tail -2
3 emp 55,000 surat
3 emp 55,000 surat
[root@localhost lab]#
```

pipe Command Example



```
[root@localhost lab]# cat f1.txt
1 abc 45,000 rajkot
1 abc 45,000 surat
4 xyz 42,000 morbi
3 emp 55,000 surat
3 emp 55,000 surat
3 emp 55,000 surat
2 pqr 33,000 ahmedabad
[root@localhost lab]# cat f1.txt | grep "abc" | grep "surat"
1 abc 45,000 surat
[root@localhost lab]#
```

tr(translate) Command

- The **tr** command in UNIX is a command line utility for translating or deleting characters.
- It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace.
- It can be used with UNIX pipes to support more complex translation.
- **tr stands for translate**.

▪ **Syntax :**

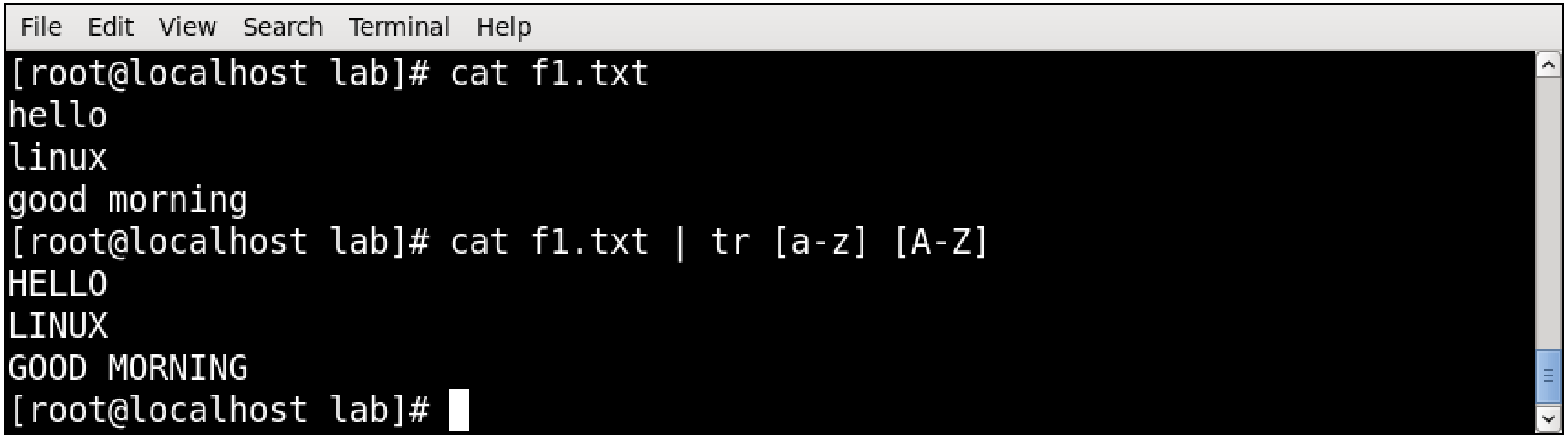
```
tr [OPTION] SET1 [SET2]
```

tr(translate) Command

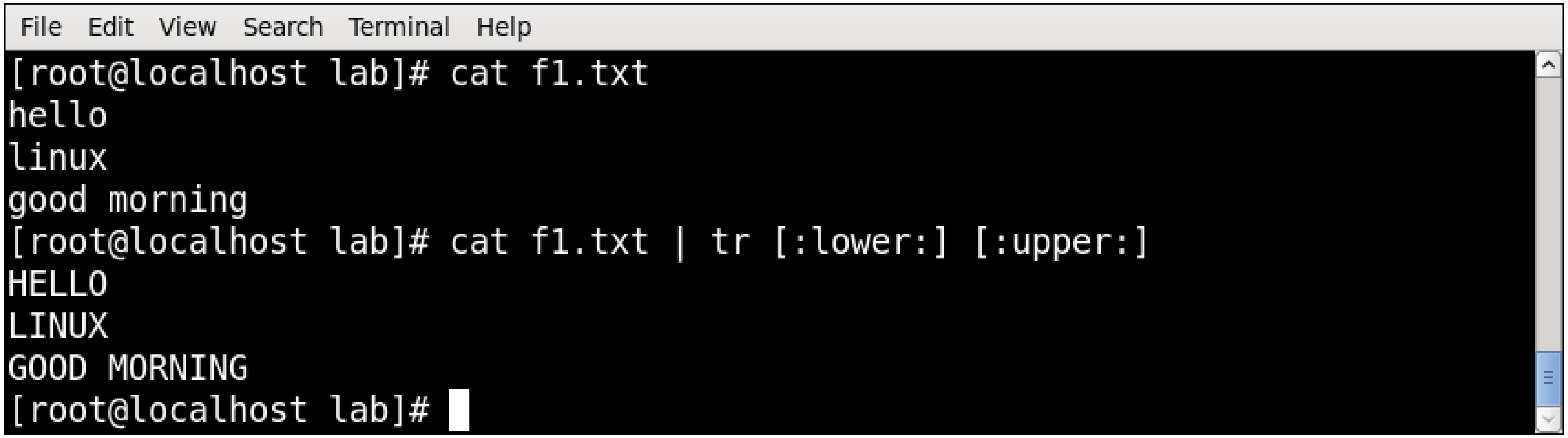
- **POSIX Character set supported by tr command :**
  - **[:digit:]** Only the digits 0 to 9.
  - **[:alnum:]** Any alphanumeric character.
  - **[:alpha:]** Any alpha character A to Z or a to z.
  - **[:blank:]** Space and TAB characters only.
  - **[:xdigit:]** Hexadecimal notation 0-9, A-F, a-f.
  - **[:upper:]** Any alpha character A to Z.
  - **[:lower:]** Any alpha character a to z..

Option	Use
tr -s	Replaces repeated characters listed in the set1 with single occurrence
tr -d	Delete characters in string1 from the input
tr -c	complements the set of characters in string. i.e., operations apply to characters not in the given set
tr -cd	Remove all characters except digits

tr Command Example

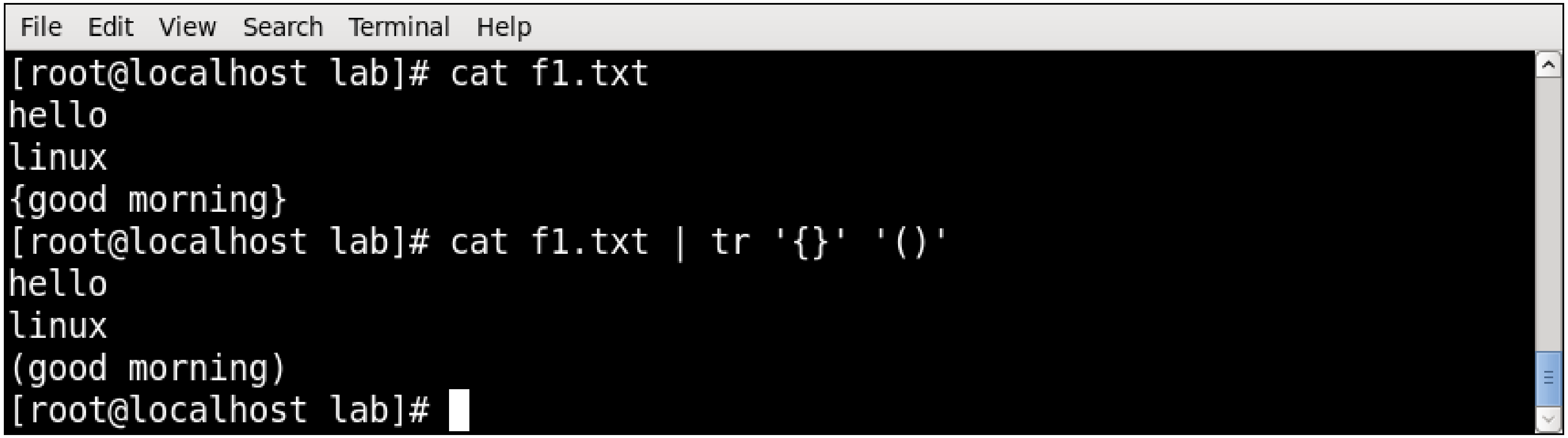


```
[root@localhost lab]# cat f1.txt
hello
linux
good morning
[root@localhost lab]# cat f1.txt | tr [a-z] [A-Z]
HELLO
LINUX
GOOD MORNING
[root@localhost lab]# 
```



```
[root@localhost lab]# cat f1.txt
hello
linux
good morning
[root@localhost lab]# cat f1.txt | tr [:lower:] [:upper:]
HELLO
LINUX
GOOD MORNING
[root@localhost lab]# 
```

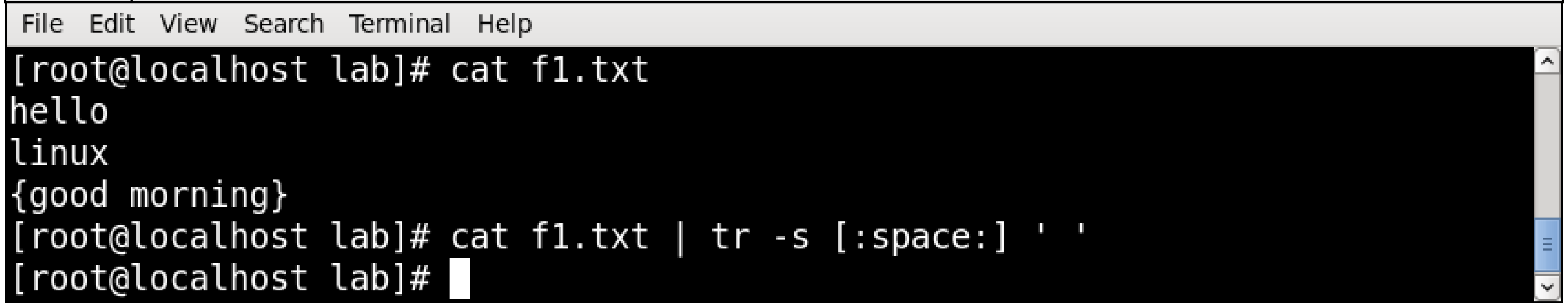
tr Command Example



```
[root@localhost lab]# cat f1.txt
hello
linux
{good morning}
[root@localhost lab]# cat f1.txt | tr '{}' '()'
hello
linux
(good morning)
[root@localhost lab]#
```

tr Command Example

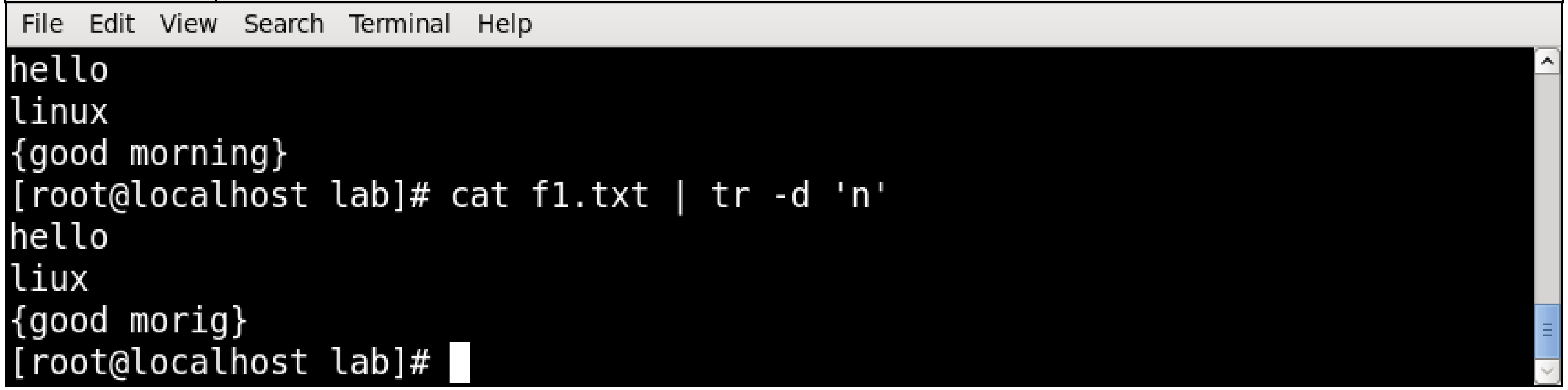
tr -s	**Replaces repeated characters listed in the set1 with single occurrence**



```
[root@localhost lab]# cat f1.txt
hello
linux
{good morning}
[root@localhost lab]# cat f1.txt | tr -s [:space:] ' '
[root@localhost lab]# 
```

tr Command Example

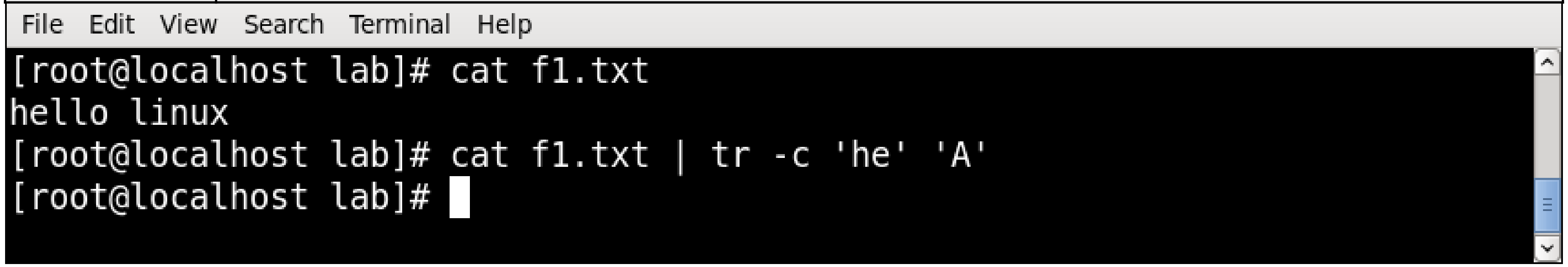
tr -d	**Delete characters in string1 from the input**



```
hello
linux
{good morning}
[root@localhost lab]# cat f1.txt | tr -d 'n'
hello
liux
{good morig}
[root@localhost lab]# 
```

tr Command Example

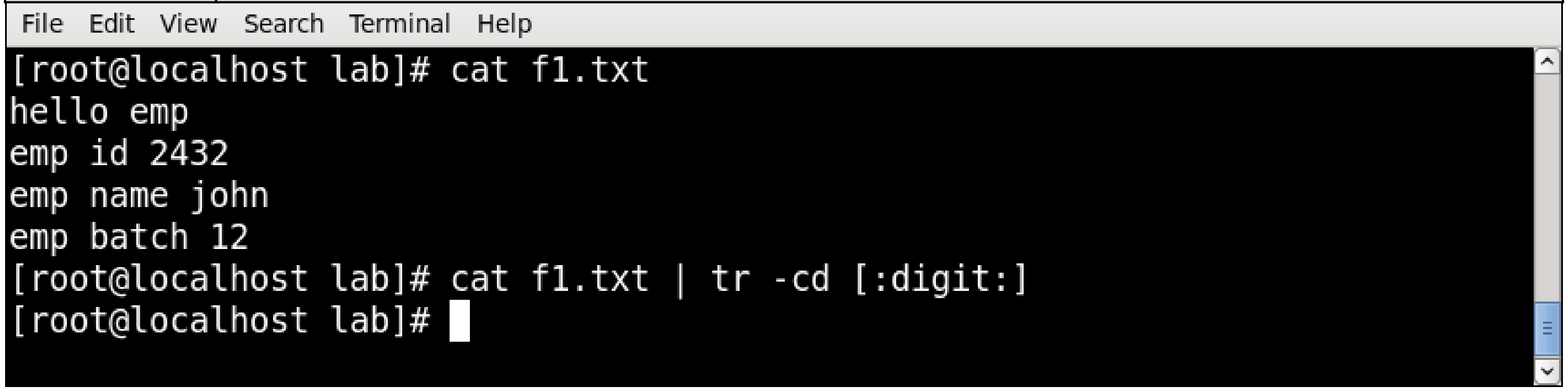
tr -c	**complements the set of characters in string.** **i.e., operations apply to characters not in the given set**



```
[root@localhost lab]# cat f1.txt
hello linux
[root@localhost lab]# cat f1.txt | tr -c 'he' 'A'
[root@localhost lab]# 
```

tr Command Example

tr -cd	**Remove all characters except digits**

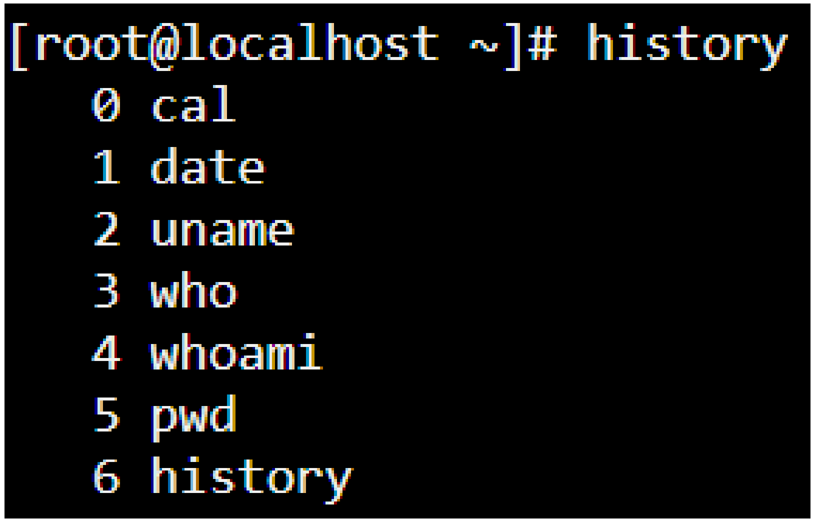


```
[root@localhost lab]# cat f1.txt
hello emp
emp id 2432
emp name john
emp batch 12
[root@localhost lab]# cat f1.txt | tr -cd [:digit:]
[root@localhost lab]#
```

history Command

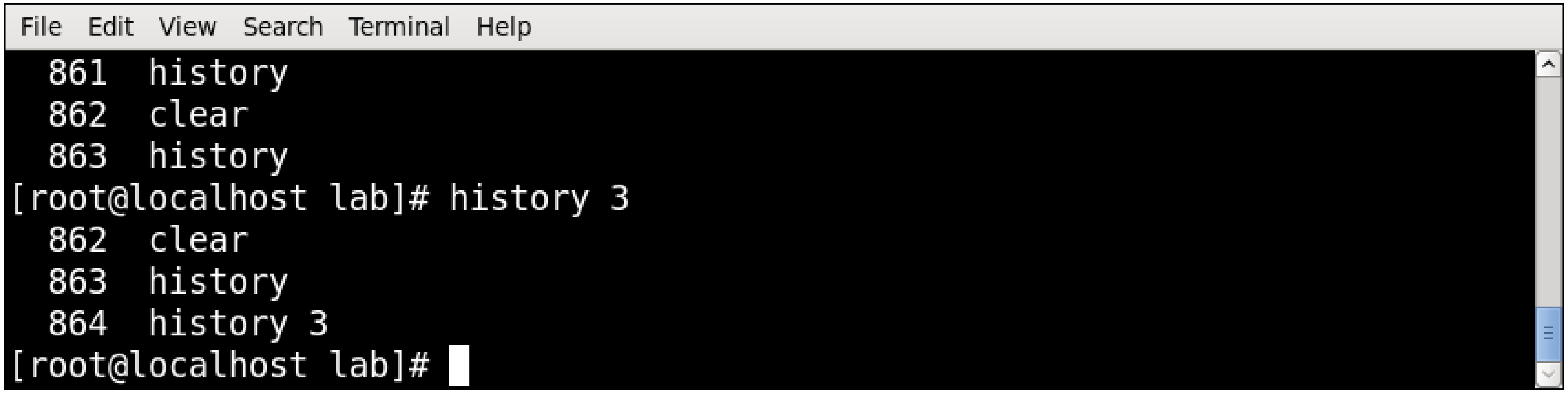
- **history** command is used to view the previously executed command.
- **Syntax :**

  history
- **Example :**

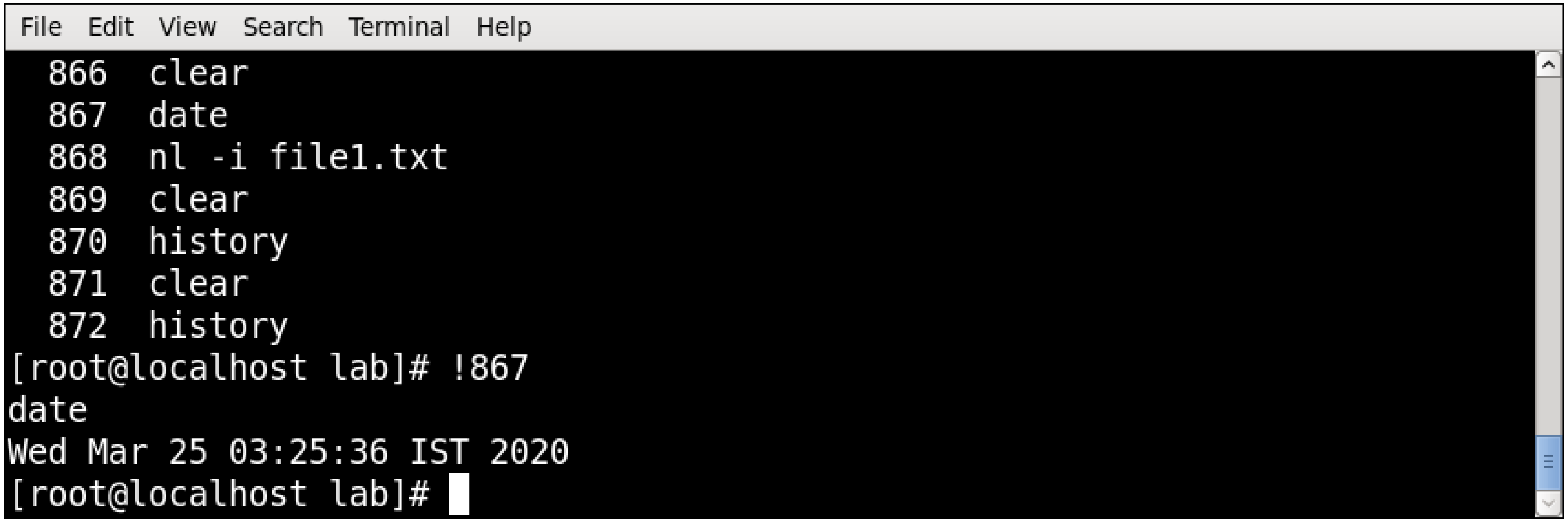


```
[root@localhost ~]# history
   0 cal
   1 date
   2 uname
   3 who
   4 whoami
   5 pwd
   6 history
```

history Command Example



```
  861  history
  862  clear
  863  history
[root@localhost lab]# history 3
  862  clear
  863  history
  864  history 3
[root@localhost lab]# 
```



```
  866  clear
  867  date
  868  nl -i file1.txt
  869  clear
  870  history
  871  clear
  872  history
[root@localhost lab]# !867
date
Wed Mar 25 03:25:36 IST 2020
[root@localhost lab]# 
```

write Command

- **write** sends a message to another user.
- **Syntax :**

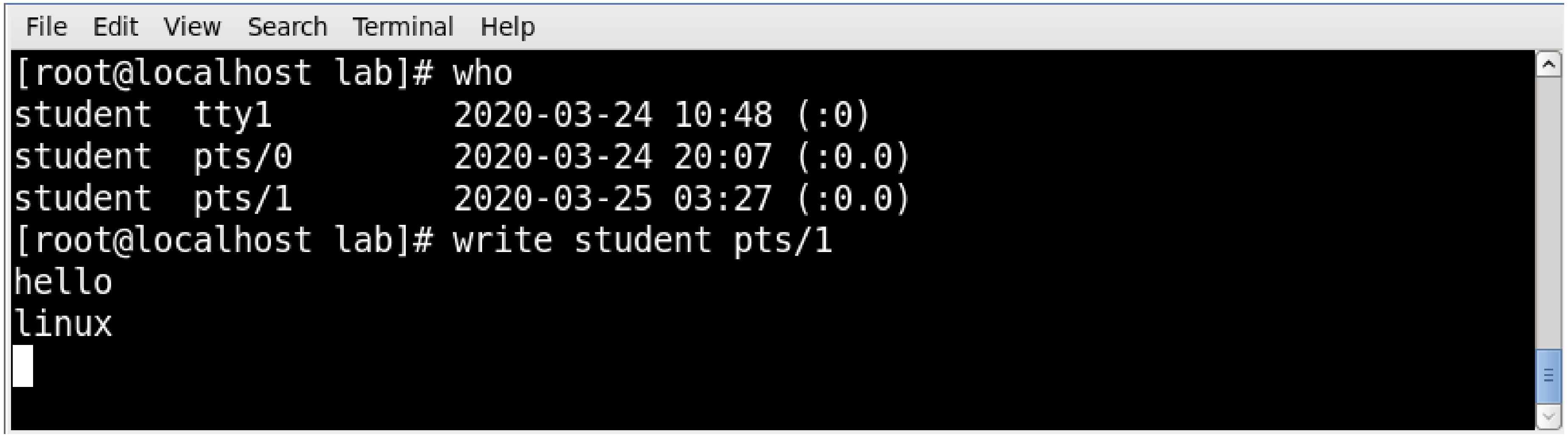
  write user [ttyname]

- **Example**

Option	Use
user	The user to write to
tty	The specific terminal to write to, if the user is logged in to more than one session

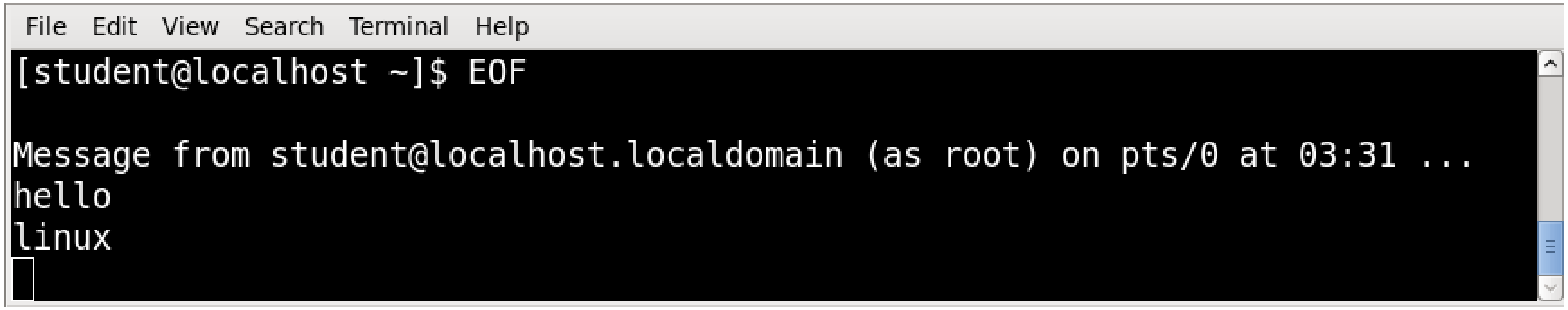
write Command Example

- Open first terminal.



```
[root@localhost lab]# who
student  tty1         2020-03-24 10:48 (:0)
student  pts/0        2020-03-24 20:07 (:0.0)
student  pts/1        2020-03-25 03:27 (:0.0)
[root@localhost lab]# write student pts/1
hello
linux
```

- Open Second terminal then execute command on first terminal.



```
[student@localhost ~]$ EOF

Message from student@localhost.localdomain (as root) on pts/0 at 03:31 ...
hello
linux
```

wall Command

- **wall** send a message to everybody’s terminal.
- wall sends a message to everybody logged in with their mesg permission set to yes.

- **Syntax :**

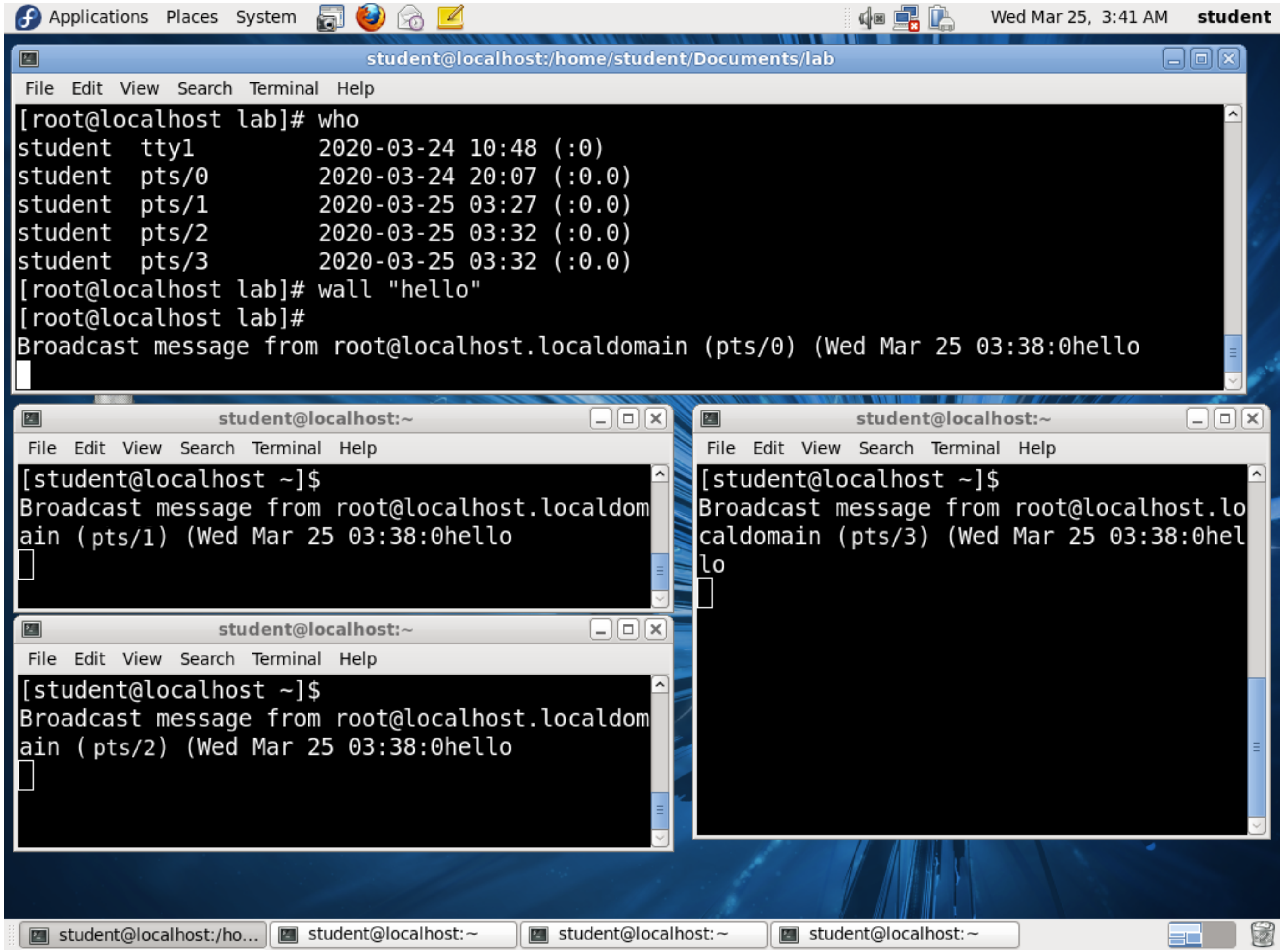
  wall [-n] [-t TIMEOUT] [file]

- **Example**

Option	Use
wall -n	--nobanner Suppress banner
wall -t	--timeout TIMEOUT Write timeout to terminals in seconds. TIMEOUT must be positive integer. Default value is 300 seconds, which is a legacy from time when people ran terminals over modem lines.

wall Command Example

- Open four different terminal, execute command on first terminal, message will display on everybody's terminal.



```
[root@localhost lab]# who
student  tty1         2020-03-24 10:48 (:0)
student  pts/0        2020-03-24 20:07 (:0.0)
student  pts/1        2020-03-25 03:27 (:0.0)
student  pts/2        2020-03-25 03:32 (:0.0)
student  pts/3        2020-03-25 03:32 (:0.0)
[root@localhost lab]# wall "hello"
[root@localhost lab]#
Broadcast message from root@localhost.localdomain (pts/0) (Wed Mar 25 03:38:0hello
```

```
[student@localhost ~]$
Broadcast message from root@localhost.localdom
ain (pts/1) (Wed Mar 25 03:38:0hello
```

```
[student@localhost ~]$
Broadcast message from root@localhost.localdom
ain (pts/2) (Wed Mar 25 03:38:0hello
```

```
[student@localhost ~]$
Broadcast message from root@localhost.lo
caldomain (pts/3) (Wed Mar 25 03:38:0hel
lo
```

Applications Places System
Wed Mar 25, 3:41 AM student
student@localhost:/home/student/Documents/lab
student@localhost:~
File Edit View Search Terminal Help
student@localhost:/ho...
student@localhost:~