cd (Change Directory) Command

- The **cd** command is used to change the current directory (i.e., the directory in which the user is currently working)

- **Syntax :**

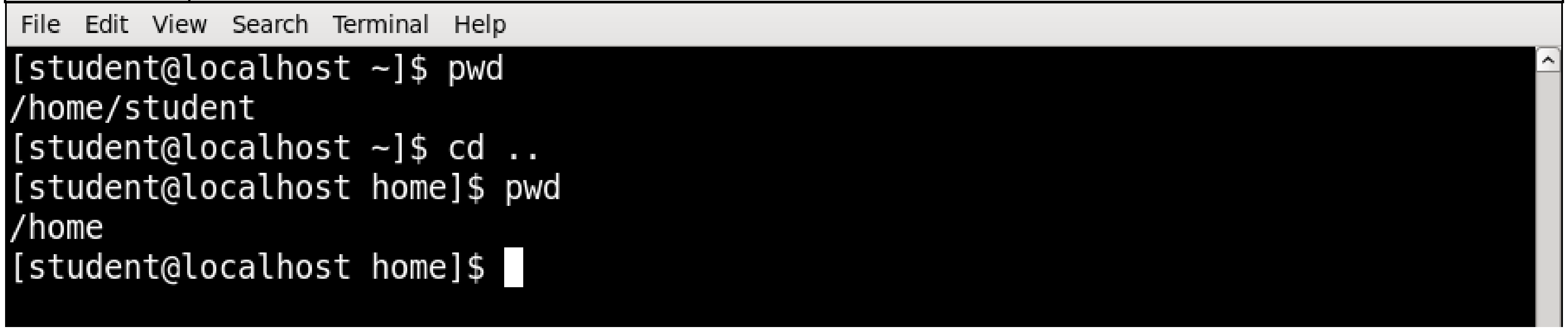
  cd [-Options] [Directory]

- **Example :**

Option	Use
cd ..	Change Current directory to parent directory
cd ~	Move to users home directory from anywhere
cd lab_1	Change from current working directory to lab_1
cd ../downloads	If we are currently in /home/username/documents then we would be placed in /home/username/downloads.

cd Command Example

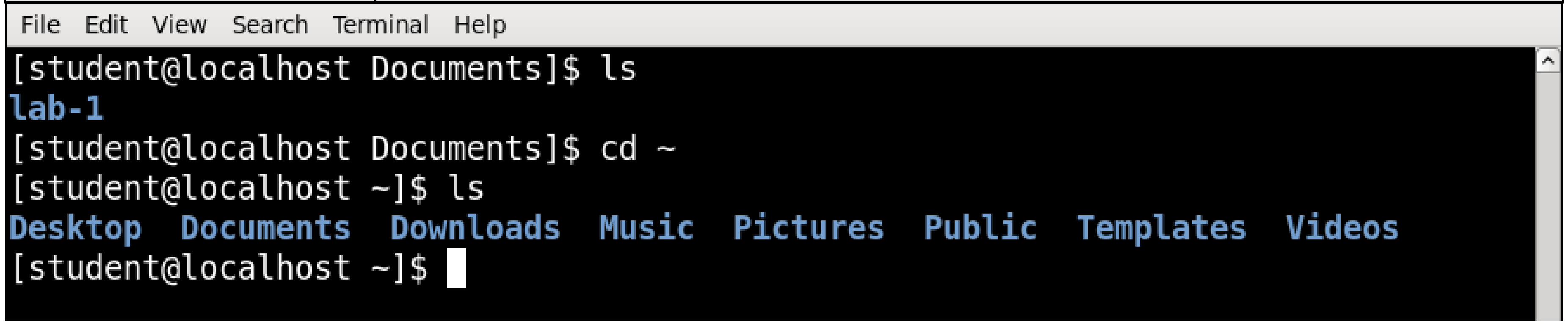
cd ..	**Change Current directory to parent directory**



```
[student@localhost ~]$ pwd
/home/student
[student@localhost ~]$ cd ..
[student@localhost home]$ pwd
/home
[student@localhost home]$
```

cd Command Example

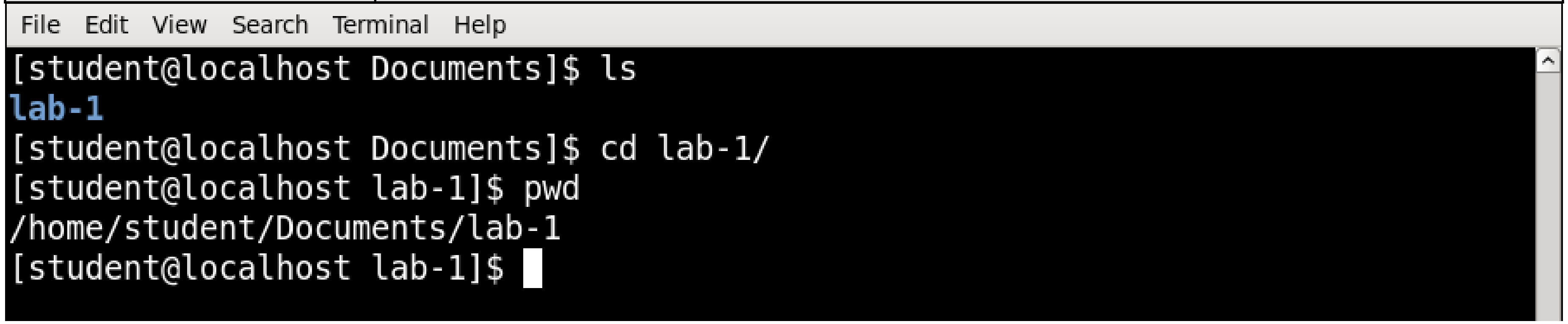
cd ~	**Move to users home directory from anywhere**



```
[student@localhost Documents]$ ls
lab-1
[student@localhost Documents]$ cd ~
[student@localhost ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[student@localhost ~]$ 
```

cd Command Example

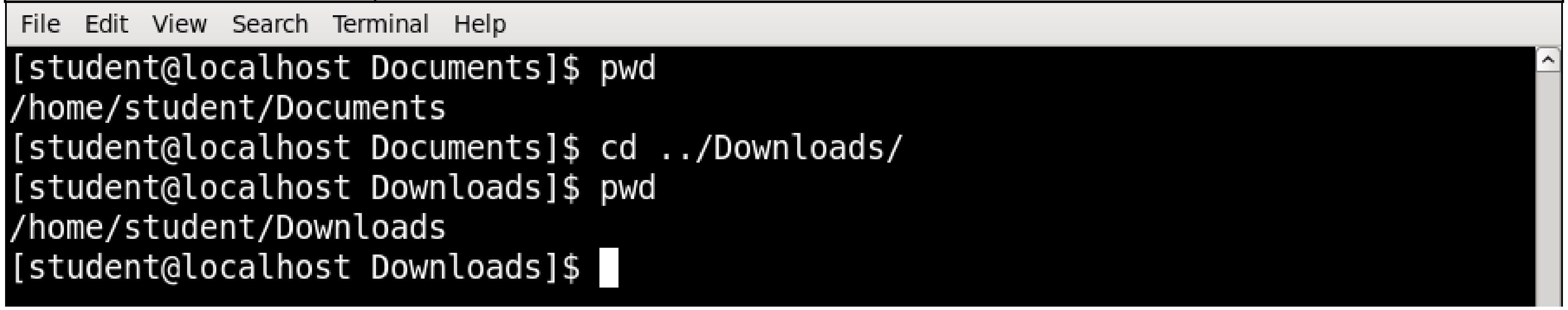
cd lab_1	**Change from current working directory to lab_1**



```
[student@localhost Documents]$ ls
lab-1
[student@localhost Documents]$ cd lab-1/
[student@localhost lab-1]$ pwd
/home/student/Documents/lab-1
[student@localhost lab-1]$ 
```

cd Command Example

cd ../downloads	**If we are currently in /home/student/documents then we would be placed in /home/student/downloads**



```
[student@localhost Documents]$ pwd
/home/student/Documents
[student@localhost Documents]$ cd ../Downloads/
[student@localhost Downloads]$ pwd
/home/student/Downloads
[student@localhost Downloads]$ 
```

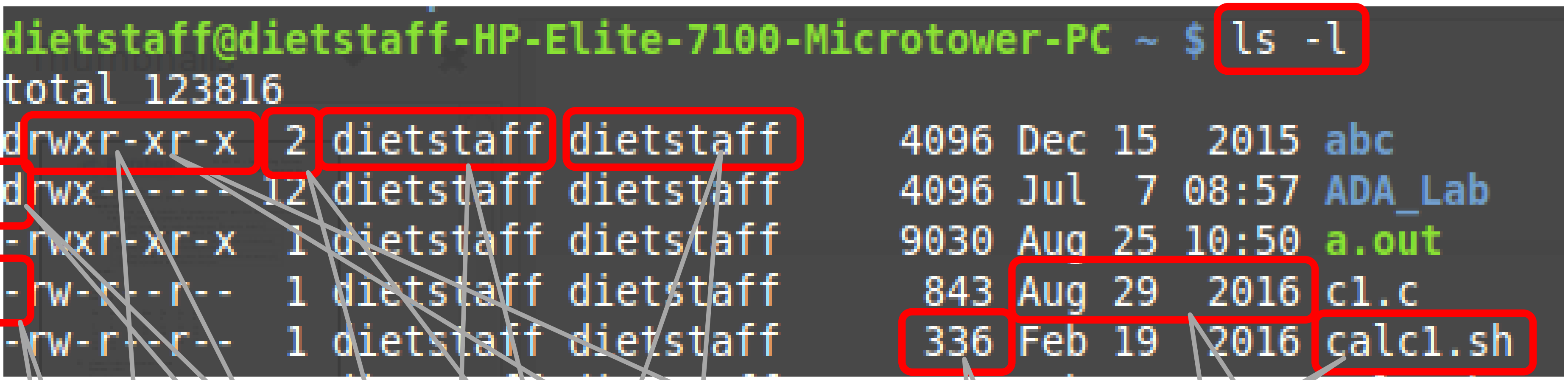
ls Command

- List directory contents.
- **Syntax :**

  ls [Options] [file|dir]
- **Example :**

Option	Use
ls -l	To show long listing information about the file/directory
ls -a	List all files including hidden file starting with '.'
ls -r	List in reverse order
ls -t	Sort by time & date
ls -s	Sort by file size

ls Command



```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ ls -l
total 123816
drwxr-xr-x  2 dietstaff dietstaff     4096 Dec 15  2015 abc
drwx------ 12 dietstaff dietstaff     4096 Jul  7 08:57 ADA_Lab
-rwxr-xr-x  1 dietstaff dietstaff     9030 Aug 25 10:50 a.out
-rw-r--r--  1 dietstaff dietstaff      843 Aug 29  2016 c1.c
-rw-r--r--  1 dietstaff dietstaff      336 Feb 19  2016 calc1.sh
```

drwxr-xr-x

$\underbrace{rwx}_{owner}\underbrace{r-x}_{group}\underbrace{r-x}_{others}$

files

directory

number of links for dir/file

read, write, execute permissions for user(root), group and others respectively

specifies owner of the file/dir

specifies group of the file/dir

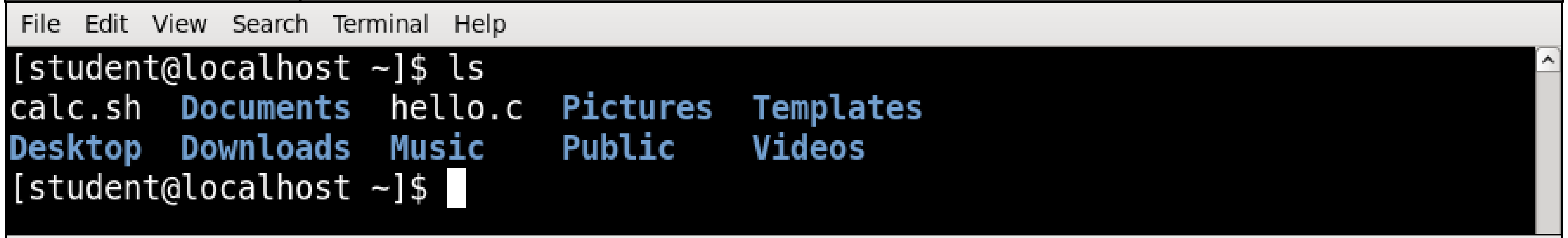
specifies the date and time of the last modification of the file

name of the file/dir

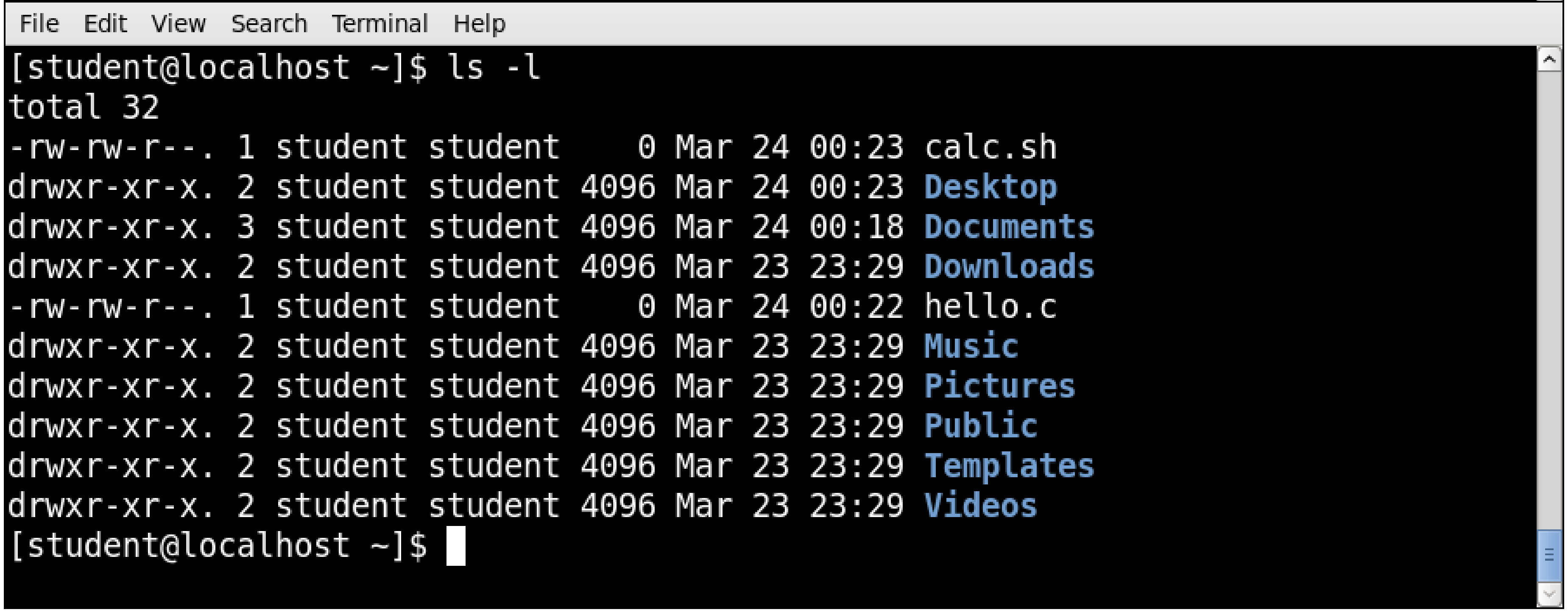
indicates the file/dir size in bytes.

ls Command Example

ls -l	**To show long listing information about the file/directory**



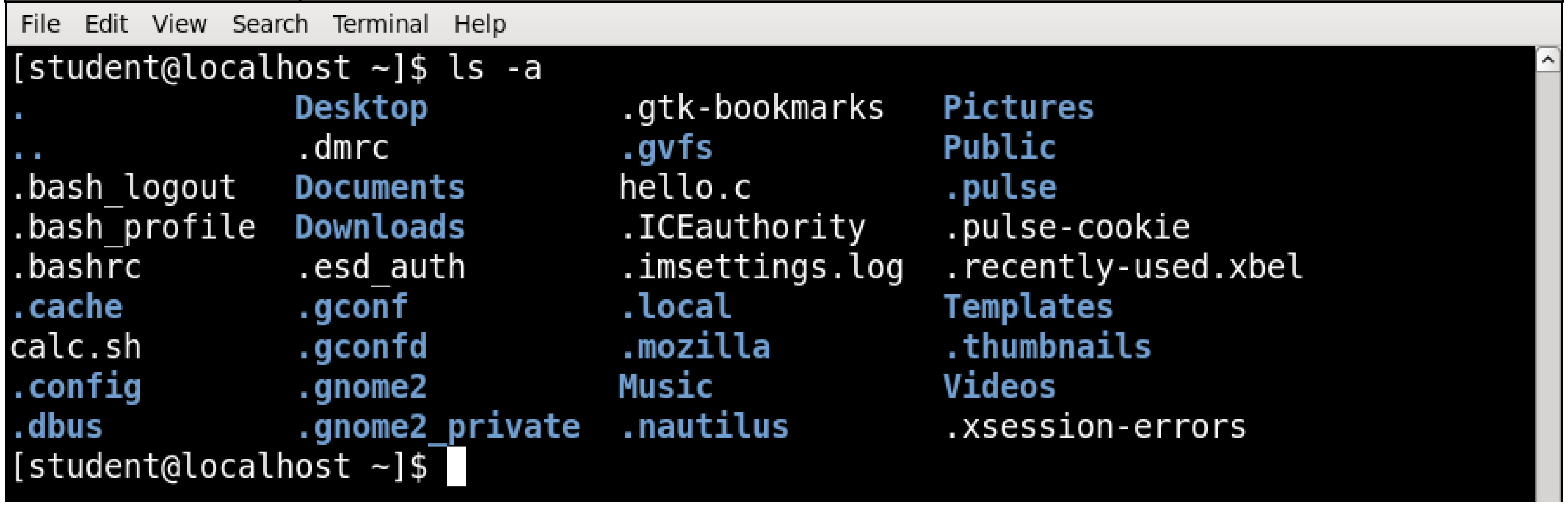
```
[student@localhost ~]$ ls
calc.sh  Documents  hello.c  Pictures  Templates
Desktop  Downloads  Music    Public    Videos
[student@localhost ~]$ 
```



```
[student@localhost ~]$ ls -l
total 32
-rw-rw-r--. 1 student student    0 Mar 24 00:23 calc.sh
drwxr-xr-x. 2 student student 4096 Mar 24 00:23 Desktop
drwxr-xr-x. 3 student student 4096 Mar 24 00:18 Documents
drwxr-xr-x. 2 student student 4096 Mar 23 23:29 Downloads
-rw-rw-r--. 1 student student    0 Mar 24 00:22 hello.c
drwxr-xr-x. 2 student student 4096 Mar 23 23:29 Music
drwxr-xr-x. 2 student student 4096 Mar 23 23:29 Pictures
drwxr-xr-x. 2 student student 4096 Mar 23 23:29 Public
drwxr-xr-x. 2 student student 4096 Mar 23 23:29 Templates
drwxr-xr-x. 2 student student 4096 Mar 23 23:29 Videos
[student@localhost ~]$ 
```

ls Command Example

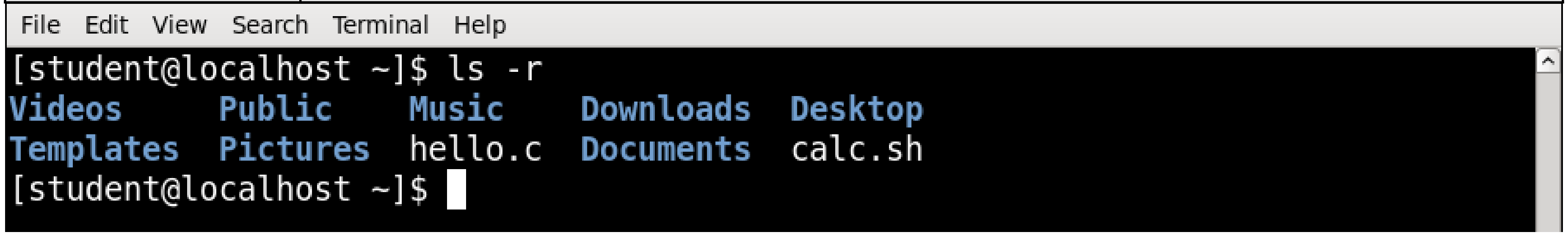
ls -a	**List all files including hidden file starting with '.'**



```
[student@localhost ~]$ ls -a
.               Desktop            .gtk-bookmarks   Pictures
..              .dmrc              .gvfs            Public
.bash_logout    Documents          hello.c          .pulse
.bash_profile   Downloads          .ICEauthority    .pulse-cookie
.bashrc         .esd_auth          .imsettings.log  .recently-used.xbel
.cache          .gconf             .local           Templates
calc.sh         .gconfd            .mozilla         .thumbnails
.config         .gnome2            Music            Videos
.dbus           .gnome2_private    .nautilus        .xsession-errors
[student@localhost ~]$
```

ls Command Example

ls -r	**List in reverse order**



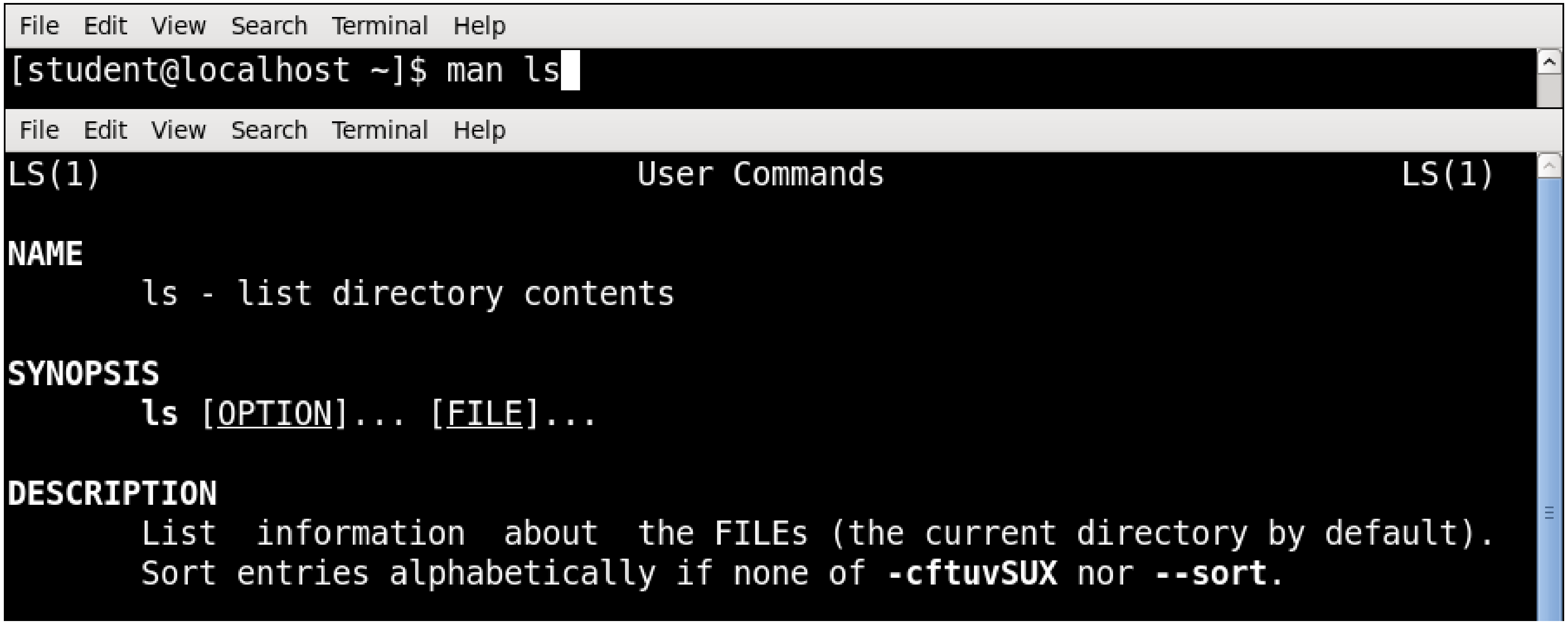
```
[student@localhost ~]$ ls -r
Videos     Public    Music    Downloads  Desktop
Templates  Pictures  hello.c  Documents  calc.sh
[student@localhost ~]$ 
```

man Command

- It is the interface used to view the system's reference manuals.
- **Syntax :**

  man [command name]

- **Example**


File Edit View Search Terminal Help

```
[student@localhost ~]$ man ls
```

File Edit View Search Terminal Help

```
LS(1)                            User Commands                            LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort.
```

echo Command

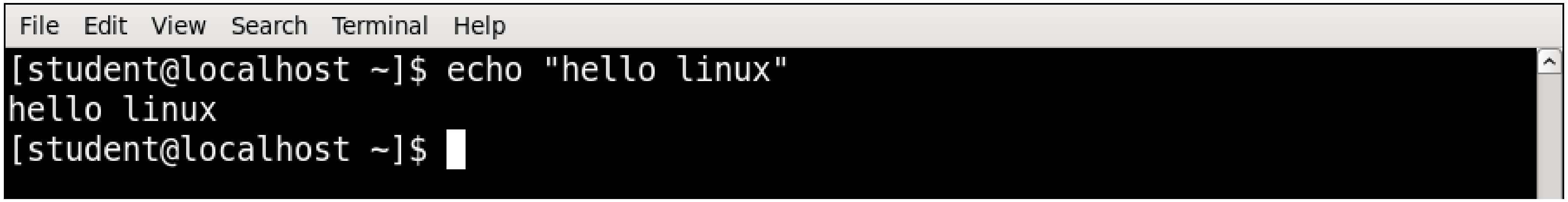
- Display a line of text/string on standard output or a file.
- **Syntax :**

  echo [option] [string]
- **Example :**

Option	Use
echo -n	Do not output a trailing newline
echo -e	Enable interpretation of backslash escape sequences

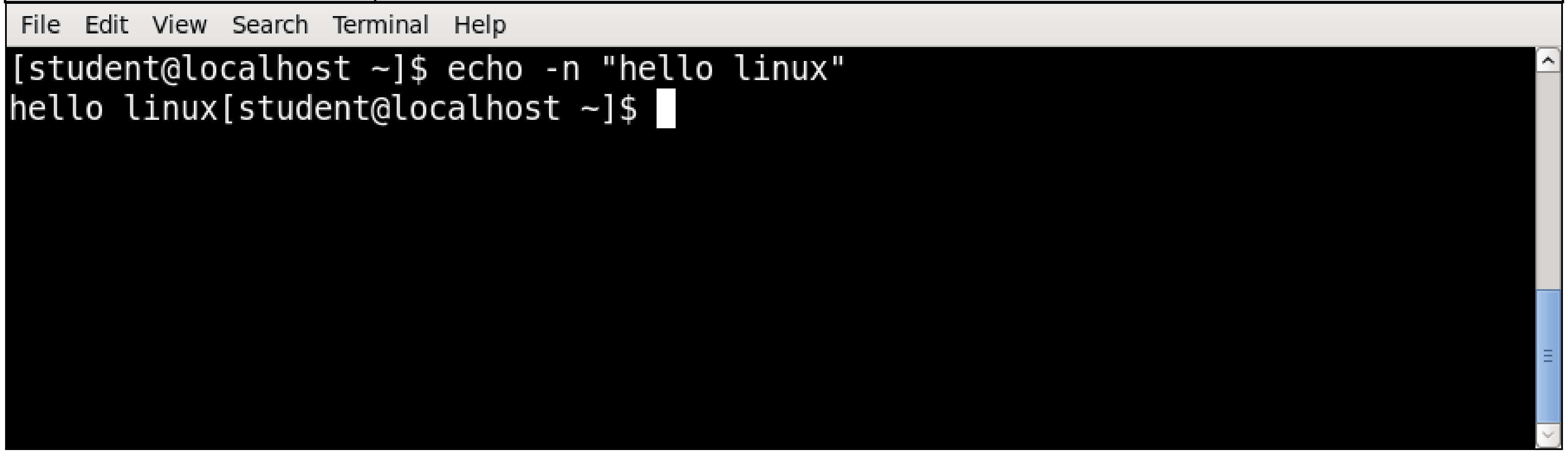
Option	Use
\b	It removes all the spaces in between the text
\n	It creates new line from where it is used
\t	It create horizontal tab spaces

echo Command Example



```
[student@localhost ~]$ echo "hello linux"
hello linux
[student@localhost ~]$ 
```

echo -n	**Do not output a trailing newline**

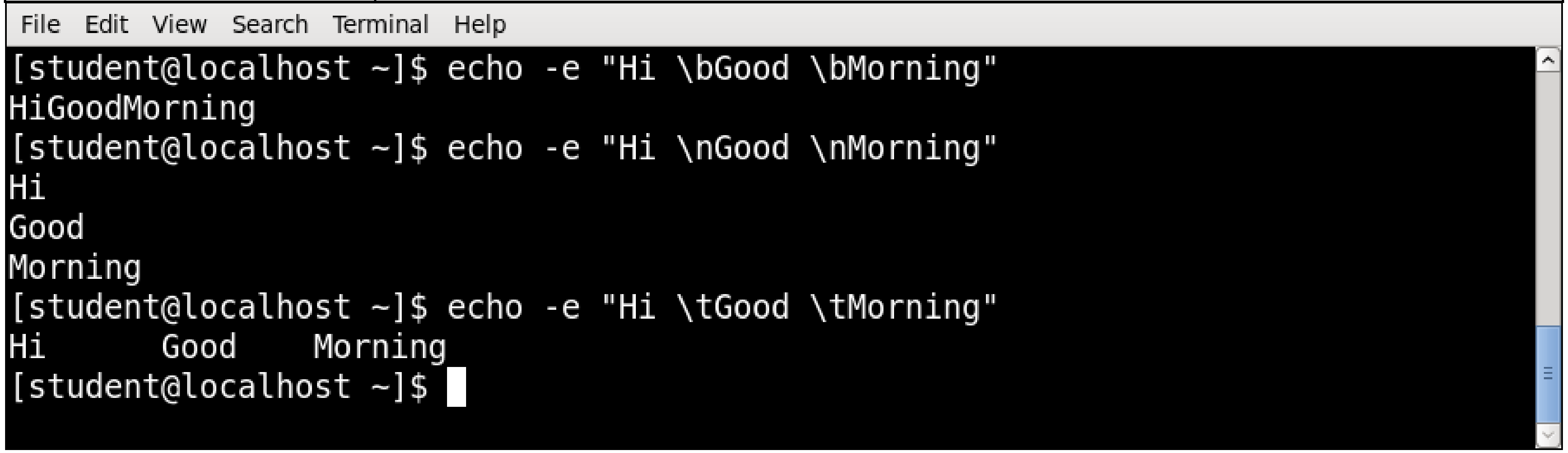


```
[student@localhost ~]$ echo -n "hello linux"
hello linux[student@localhost ~]$ 
```

echo Command Example

echo -e	Enable interpretation of backslash escape sequences

\b	It removes all the spaces in between the text
\n	It creates new line from where it is used
\t	It create horizontal tab spaces



```
[student@localhost ~]$ echo -e "Hi \bGood \bMorning"
HiGoodMorning
[student@localhost ~]$ echo -e "Hi \nGood \nMorning"
Hi
Good
Morning
[student@localhost ~]$ echo -e "Hi \tGood \tMorning"
Hi      Good    Morning
[student@localhost ~]$ 
```

cal Command

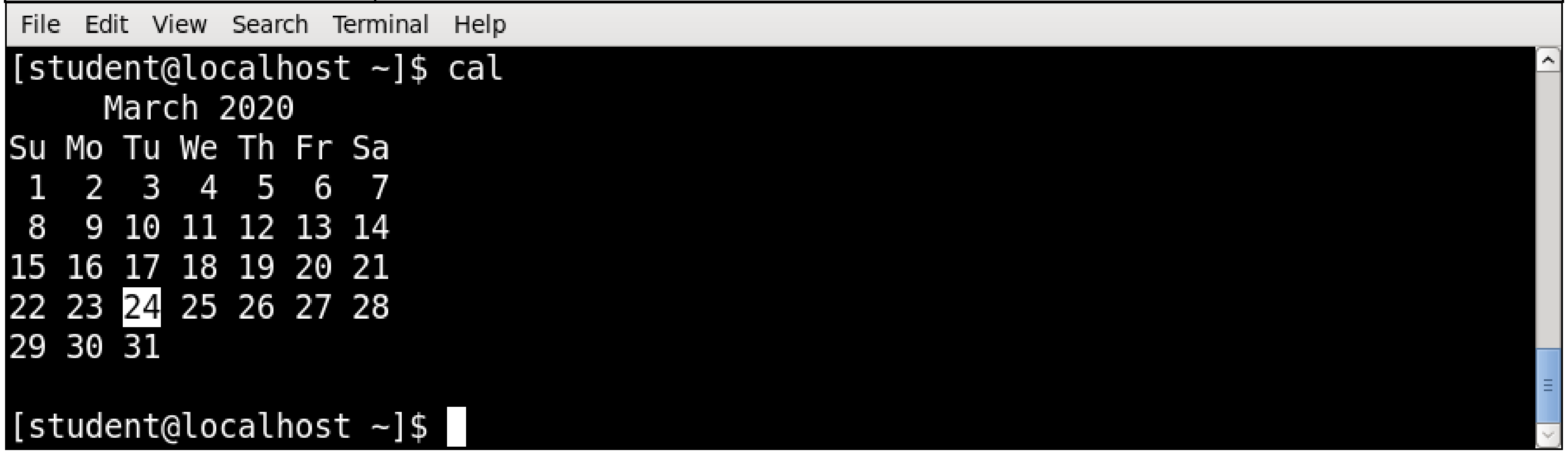
- Displays a simple, formatted calendar in your terminal.
- **Syntax :**

  cal [options] [[[day] month] year]
- **Example :**

Option	Use
cal -1	Display single month output. (This is the default.)
cal -3	Display three months spanning the date.
cal -s	Display Sunday as the first day of the week.
cal -m	Display Monday as the first day of the week.
cal -j	Use day-of-year numbering for all calendars. These are also called ordinal days. Ordinal days range from 1 to 366.
cal –y	Display a calendar for the whole year

cal Command Example

Cal or cal -1	**Display single month output. (This is the default.)**

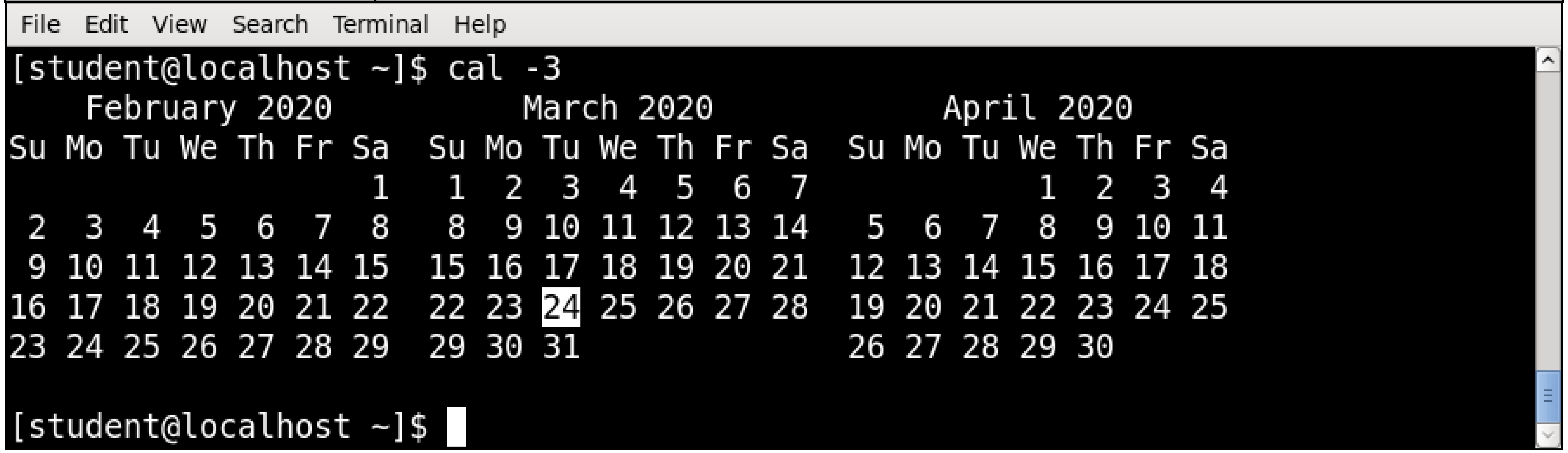


```
[student@localhost ~]$ cal
     March 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

[student@localhost ~]$
```

cal Command Example

cal -3	**Display three months spanning the date.**

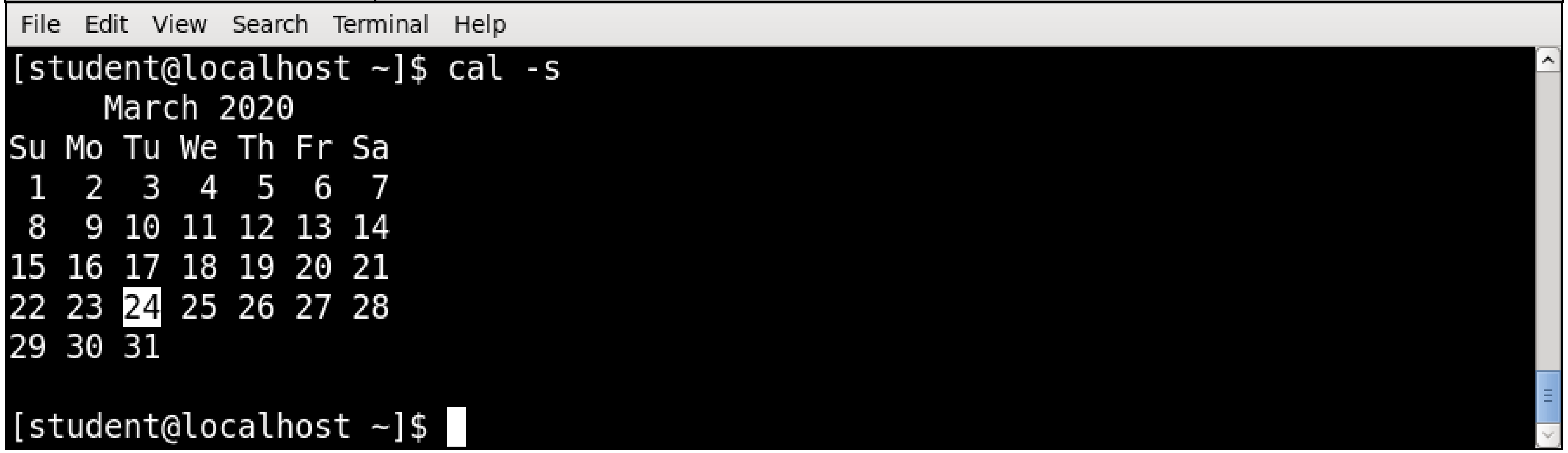


```
[student@localhost ~]$ cal -3
    February 2020          March 2020            April 2020
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
                   1   1  2  3  4  5  6  7            1  2  3  4
 2  3  4  5  6  7  8   8  9 10 11 12 13 14   5  6  7  8  9 10 11
 9 10 11 12 13 14 15  15 16 17 18 19 20 21  12 13 14 15 16 17 18
16 17 18 19 20 21 22  22 23 24 25 26 27 28  19 20 21 22 23 24 25
23 24 25 26 27 28 29  29 30 31              26 27 28 29 30

[student@localhost ~]$ 
```

cal Command Example

cal -s	**Display Sunday as the first day of the week.**

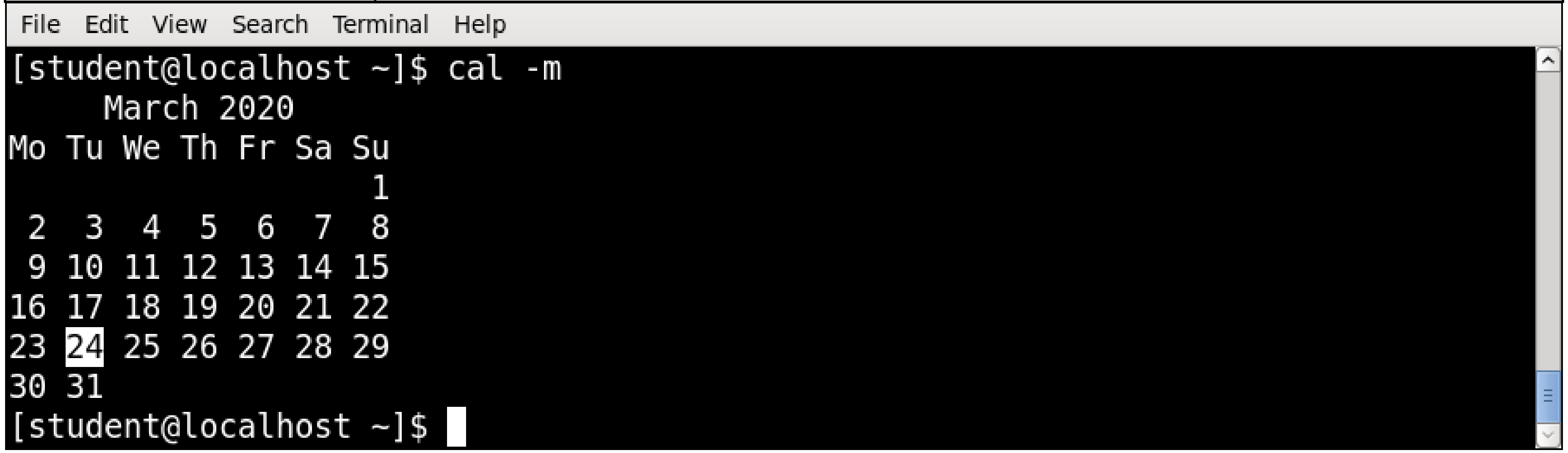


```
[student@localhost ~]$ cal -s
     March 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

[student@localhost ~]$ 
```

cal Command Example

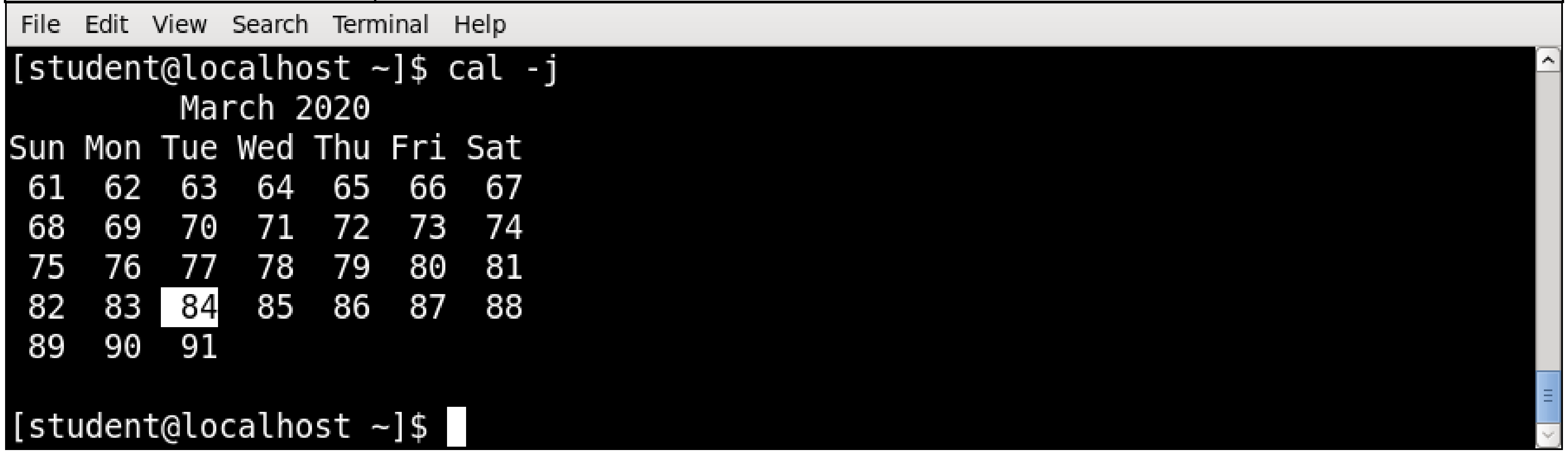
cal -m	**Display Monday as the first day of the week.**



```
[student@localhost ~]$ cal -m
     March 2020
Mo Tu We Th Fr Sa Su
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
[student@localhost ~]$
```

cal Command Example

cal -j	**Use day-of-year numbering for all calendars. These are also called ordinal days. Ordinal days range from 1 to 366.**

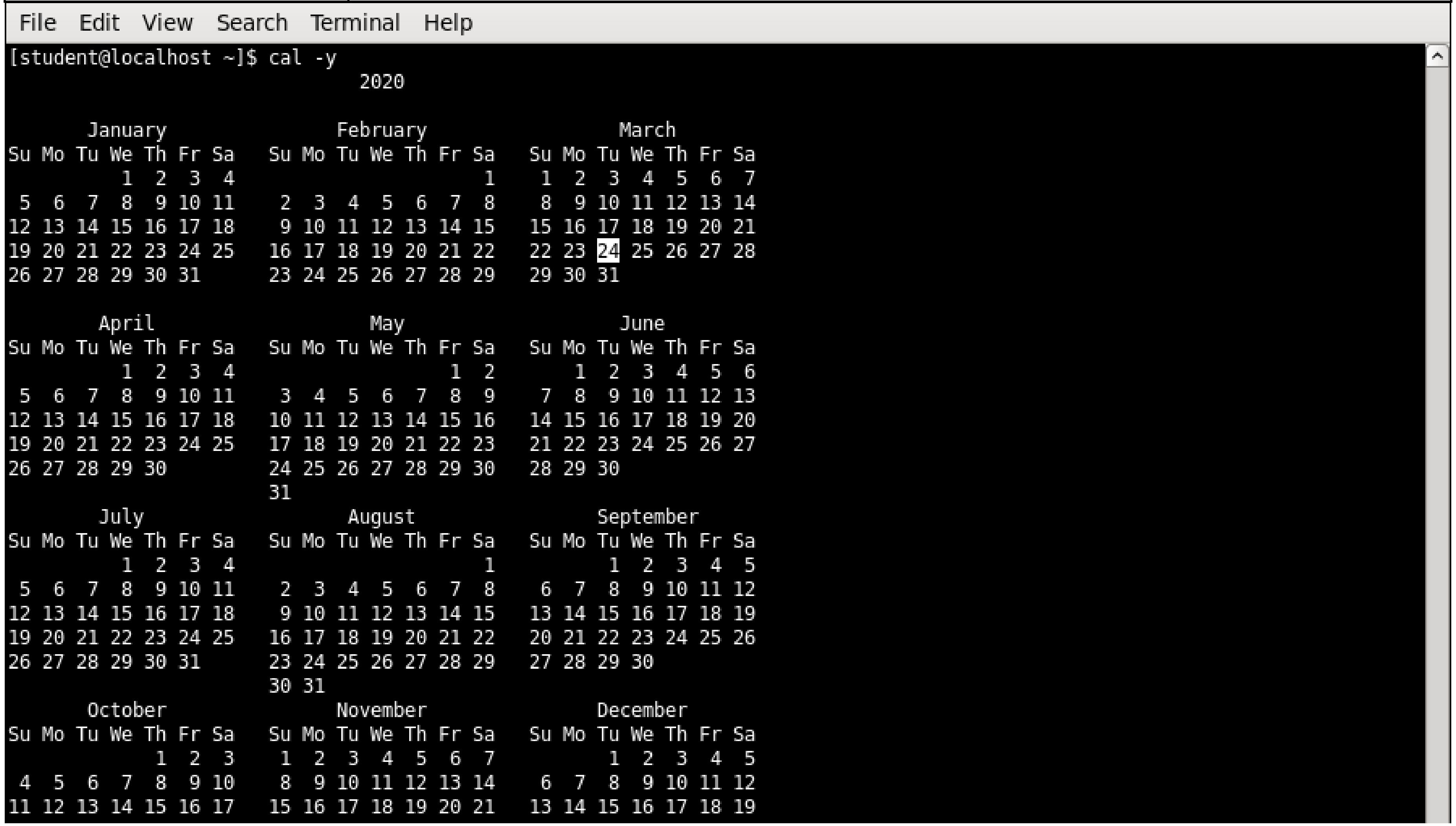


```
[student@localhost ~]$ cal -j
         March 2020
Sun Mon Tue Wed Thu Fri Sat
 61  62  63  64  65  66  67
 68  69  70  71  72  73  74
 75  76  77  78  79  80  81
 82  83  84  85  86  87  88
 89  90  91

[student@localhost ~]$ 
```

cal Command Example

cal –y	**Display a calendar for the whole year**



```
[student@localhost ~]$ cal -y
                               2020

       January               February                 March
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
          1  2  3  4                      1    1  2  3  4  5  6  7
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    8  9 10 11 12 13 14
12 13 14 15 16 17 18    9 10 11 12 13 14 15   15 16 17 18 19 20 21
19 20 21 22 23 24 25   16 17 18 19 20 21 22   22 23 24 25 26 27 28
26 27 28 29 30 31      23 24 25 26 27 28 29   29 30 31

        April                   May                   June
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
          1  2  3  4                   1  2       1  2  3  4  5  6
 5  6  7  8  9 10 11    3  4  5  6  7  8  9    7  8  9 10 11 12 13
12 13 14 15 16 17 18   10 11 12 13 14 15 16   14 15 16 17 18 19 20
19 20 21 22 23 24 25   17 18 19 20 21 22 23   21 22 23 24 25 26 27
26 27 28 29 30         24 25 26 27 28 29 30   28 29 30
                       31
        July                  August                September
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
          1  2  3  4                      1          1  2  3  4  5
 5  6  7  8  9 10 11    2  3  4  5  6  7  8    6  7  8  9 10 11 12
12 13 14 15 16 17 18    9 10 11 12 13 14 15   13 14 15 16 17 18 19
19 20 21 22 23 24 25   16 17 18 19 20 21 22   20 21 22 23 24 25 26
26 27 28 29 30 31      23 24 25 26 27 28 29   27 28 29 30
                       30 31
       October               November               December
Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa   Su Mo Tu We Th Fr Sa
             1  2  3    1  2  3  4  5  6  7          1  2  3  4  5
 4  5  6  7  8  9 10    8  9 10 11 12 13 14    6  7  8  9 10 11 12
11 12 13 14 15 16 17   15 16 17 18 19 20 21   13 14 15 16 17 18 19
```

date Command

- Print or set the system date and time.
- **Syntax :**

  date [OPTION]... [+FORMAT]
- **Example :**

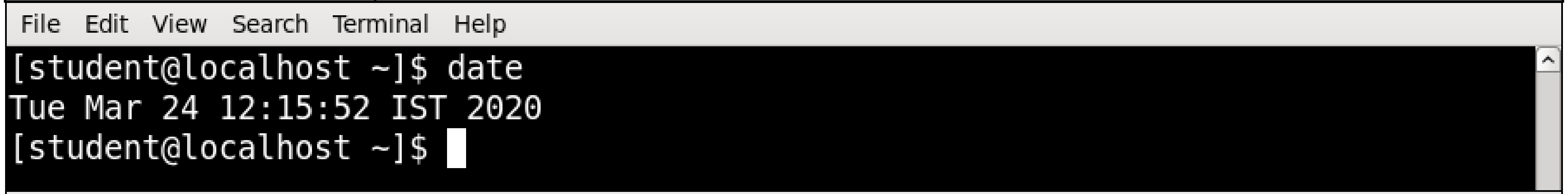
Option	Use
date +%a	The abbreviated weekday name (e.g., Sun)
date +%A	The full weekday name (e.g., Sunday)
date +%b	The abbreviated month name (e.g., Jan)
date +%B	Locale's full month name (e.g., January)
date +%C	The current century; like %Y, except omit last two digits (e.g., 20)
date +%w	day of week (0..6); 0 is Sunday

date Command

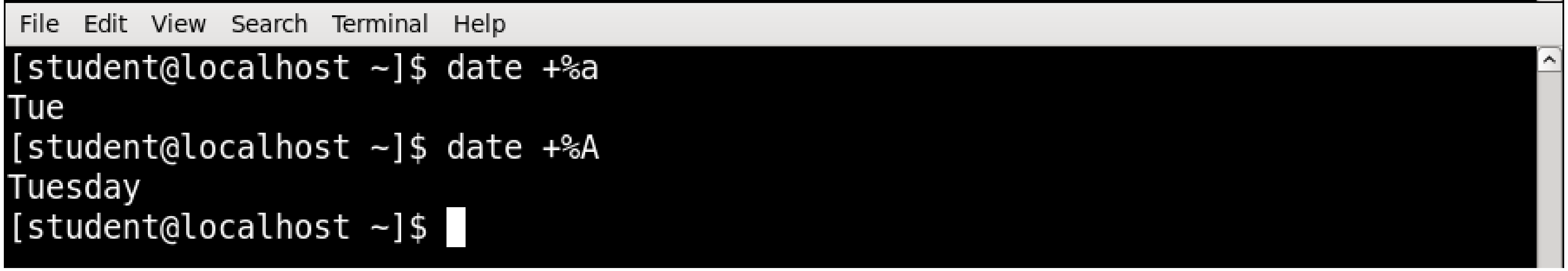
Option	Use
date +%d	Display the day of the month
date +%m	Displays the month of year (01 to 12)
date +%y	Displays last two digits of the year(00 to 99)
date +%Y	Display four-digit year.
date +%T	Display the time in 24 hour format as HH:MM:SS
date +%H	Display the hour
date +%M	Display the minute
date +%S	Display the seconds
date +%V	ISO week number, with Monday as first day of week (01..53)
date +%P	locale's equivalent of either AM or PM

date Command Example

date +%a	The abbreviated weekday name (e.g., Sun)
date +%A	The full weekday name (e.g., Sunday)



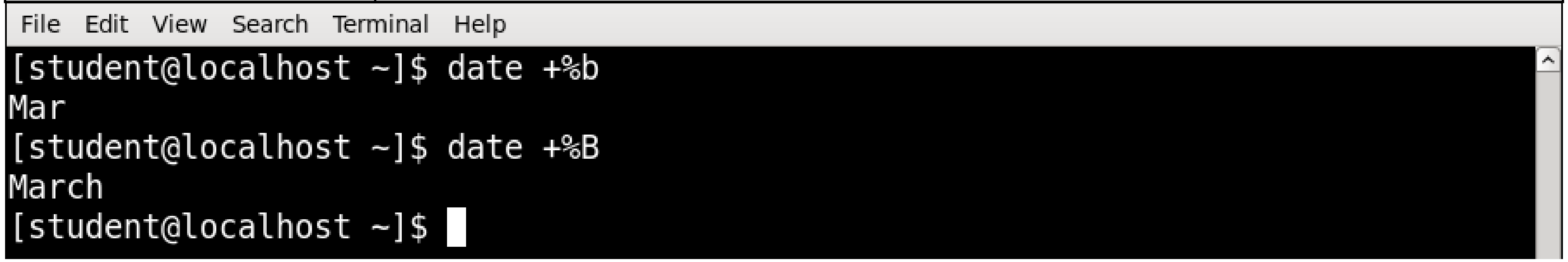
```
[student@localhost ~]$ date
Tue Mar 24 12:15:52 IST 2020
[student@localhost ~]$ 
```



```
[student@localhost ~]$ date +%a
Tue
[student@localhost ~]$ date +%A
Tuesday
[student@localhost ~]$ 
```

date Command Example

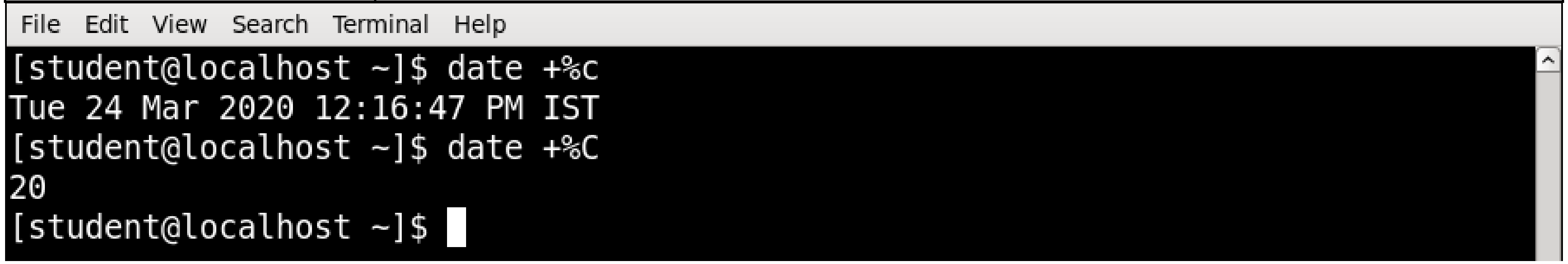
date +%b	**The abbreviated month name (e.g., Jan)**
date +%B	**Locale's full month name (e.g., January)**



```
[student@localhost ~]$ date +%b
Mar
[student@localhost ~]$ date +%B
March
[student@localhost ~]$ 
```

date Command Example

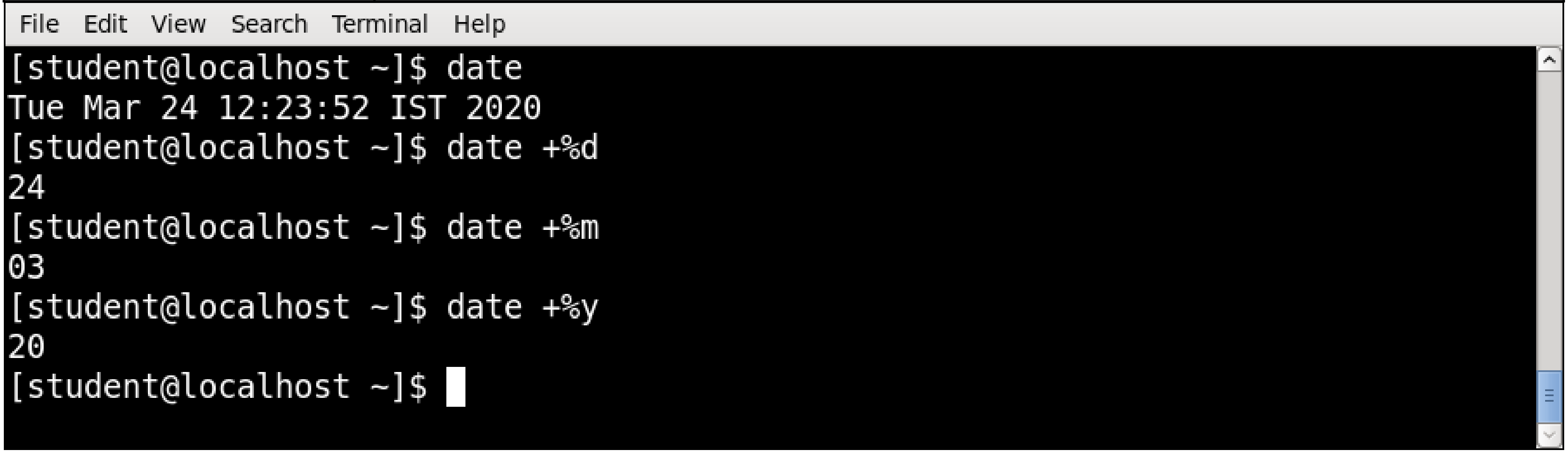
date +%c	**Full date with IST timing**
date +%C	**The current century; like %Y, except omit last two digits (e.g., 20)**



```
[student@localhost ~]$ date +%c
Tue 24 Mar 2020 12:16:47 PM IST
[student@localhost ~]$ date +%C
20
[student@localhost ~]$
```

date Command Example

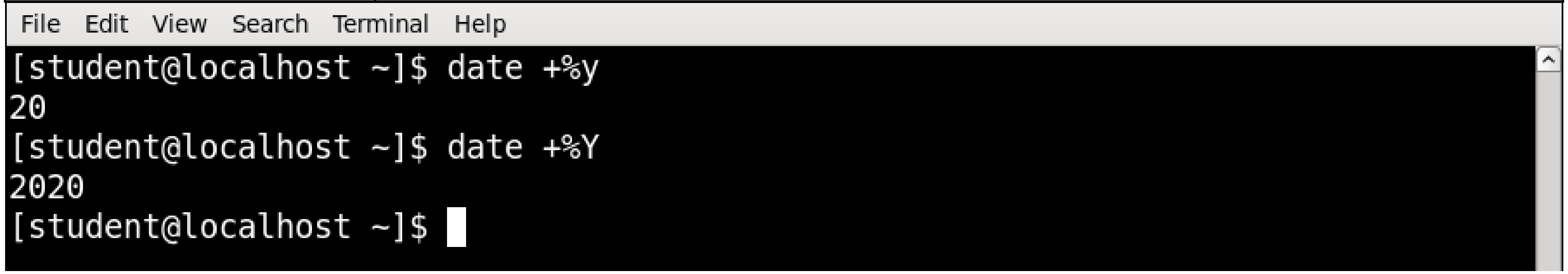
date +%d	**Display the day of the month**
date +%m	**Displays the month of year (01 to 12)**
date +%y	**Displays last two digits of the year(00 to 99)**



```
[student@localhost ~]$ date
Tue Mar 24 12:23:52 IST 2020
[student@localhost ~]$ date +%d
24
[student@localhost ~]$ date +%m
03
[student@localhost ~]$ date +%y
20
[student@localhost ~]$ 
```

date Command Example

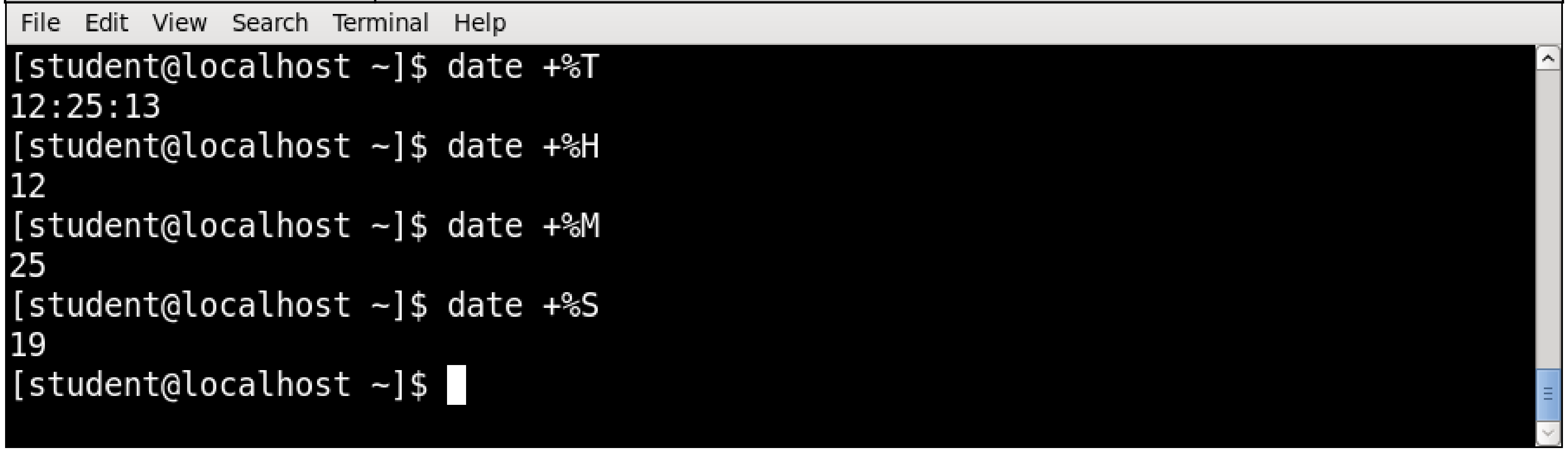
date +%y	**Displays last two digits of the year(00 to 99)**
date +%Y	**Display four-digit year.**



```
[student@localhost ~]$ date +%y
20
[student@localhost ~]$ date +%Y
2020
[student@localhost ~]$ 
```

date Command Example

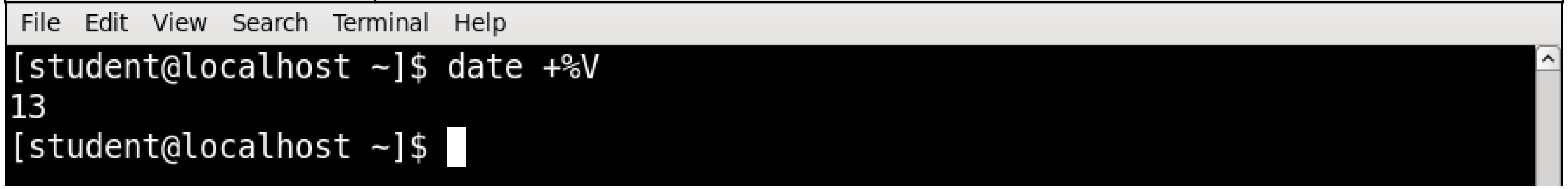
date +%T	**Display the time in 24 hour format as HH:MM:SS**
date +%H	**Display the hour**
date +%M	**Display the minute**
date +%S	**Display the seconds**



```
[student@localhost ~]$ date +%T
12:25:13
[student@localhost ~]$ date +%H
12
[student@localhost ~]$ date +%M
25
[student@localhost ~]$ date +%S
19
[student@localhost ~]$ 
```

date Command Example

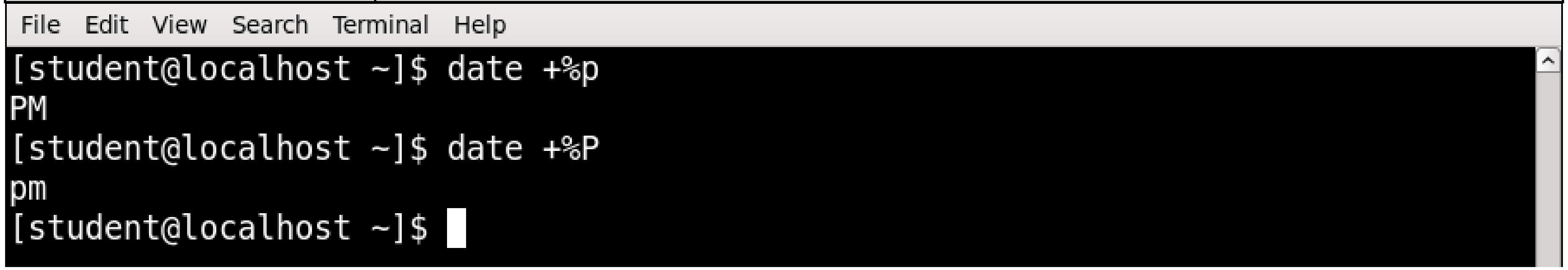
date +%V	**ISO week number, with Monday as first day of week (01..53)**



```
[student@localhost ~]$ date +%V
13
[student@localhost ~]$
```

date Command Example

date +%P	**locale's equivalent of either AM or PM**



```
[student@localhost ~]$ date +%p
PM
[student@localhost ~]$ date +%P
pm
[student@localhost ~]$
```

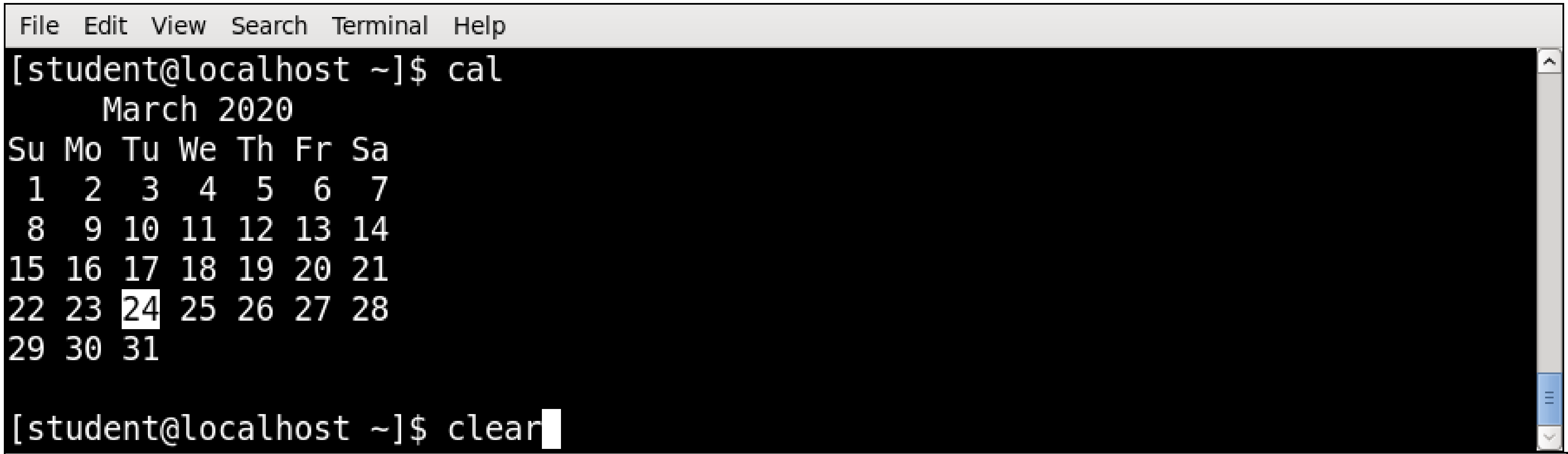
clear Command

- Clear the terminal screen.
- If you take a detailed look after running the clear command, you'll find that it doesn't really clear the terminal. The tool just shifts the text upwards, out of the viewable area.

▪ **Syntax :**

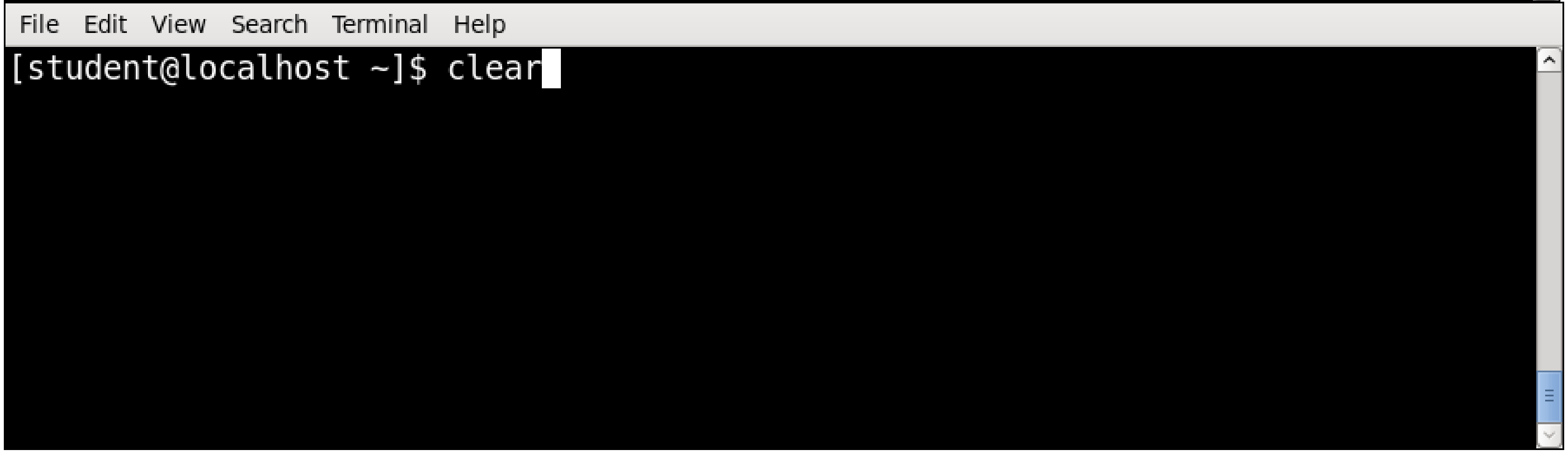
```
clear
```

clear Command Example



```
[student@localhost ~]$ cal
     March 2020
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31

[student@localhost ~]$ clear
```



```
[student@localhost ~]$ clear
```

cat Command

- It is used to create, display and concatenate file contents.
- **Syntax :**

  cat [OPTION] [FILE]
- **Example :**

Option	Use
cat -b	Omits line numbers for blank space in the output
cat -E	Displays a $ (dollar sign) at the end of each line
cat -n	Line numbers for all the output lines
cat -s	Suppress repeated empty output lines
cat -T	Displays the tab characters as ^I in the output

cat Command

- **Example :**

  $ cat > file1.txt

- It creates file1.txt and allow us to insert content for this file.
- After inserting content you can use ctrl+c to exit the file.

  $cat file.txt > newfile.txt

- Read the contents of file.txt and write them to newfile.txt, overwriting anything newfile.txt previously contained. If newfile.txt does not exist, it will be created.

  $cat file.txt >> newfile.txt

- Read the contents of **file.txt** and append them to the end of **newfile.txt**. If **newfile.txt** does not exist, it will be created.

cat Command

- **Example :**

  cat file1.txt file2.txt

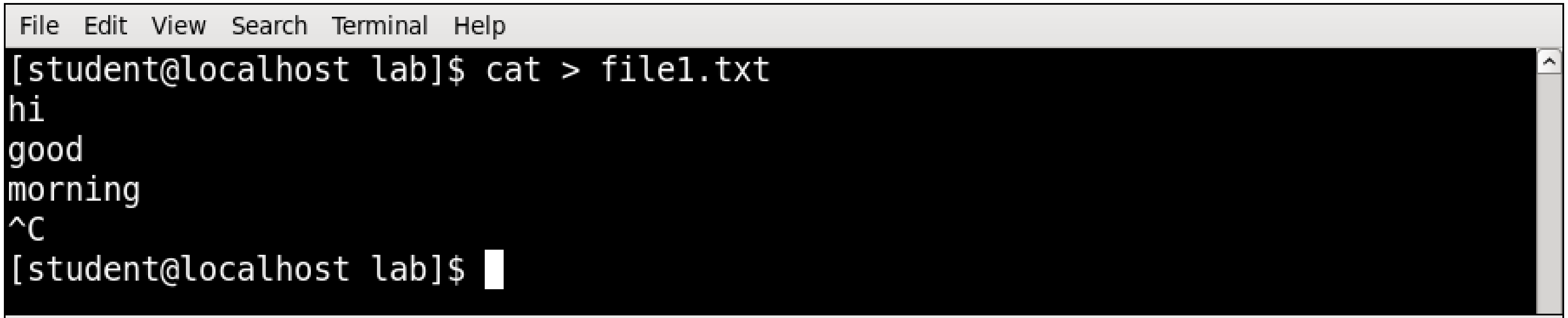
- It will read the contents of file1.txt and file2.txt and display the result in the terminal.

  cat file1.txt file2.txt > combinedfile.txt

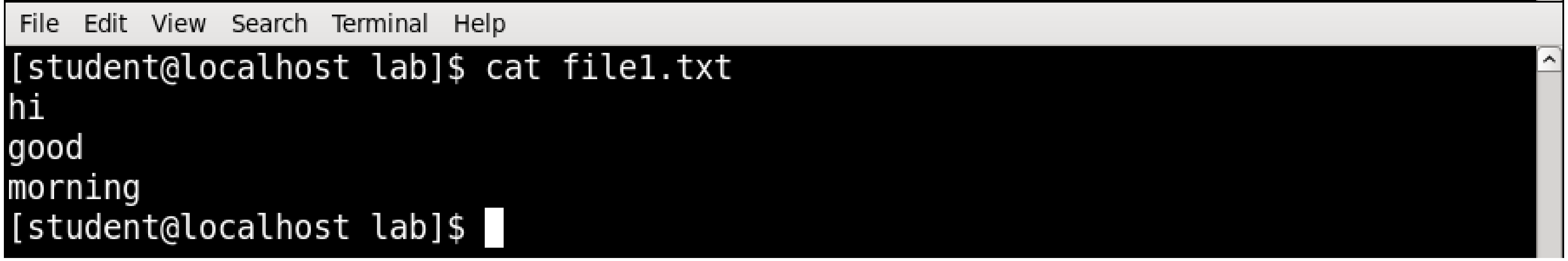
- It will concatenate the contents of file1.txt and file2.txt and write them to a new file combinedfile.txt using the (>) operator.
- If the combinedfile.txt file doesn’t exist the command will create it. Otherwise it will overwrite the file.

cat Command Example

- **$ cat > file1.txt**
- It creates file1.txt and allow us to insert content for this file.
- After inserting content you can use ctrl+c to exit the file.



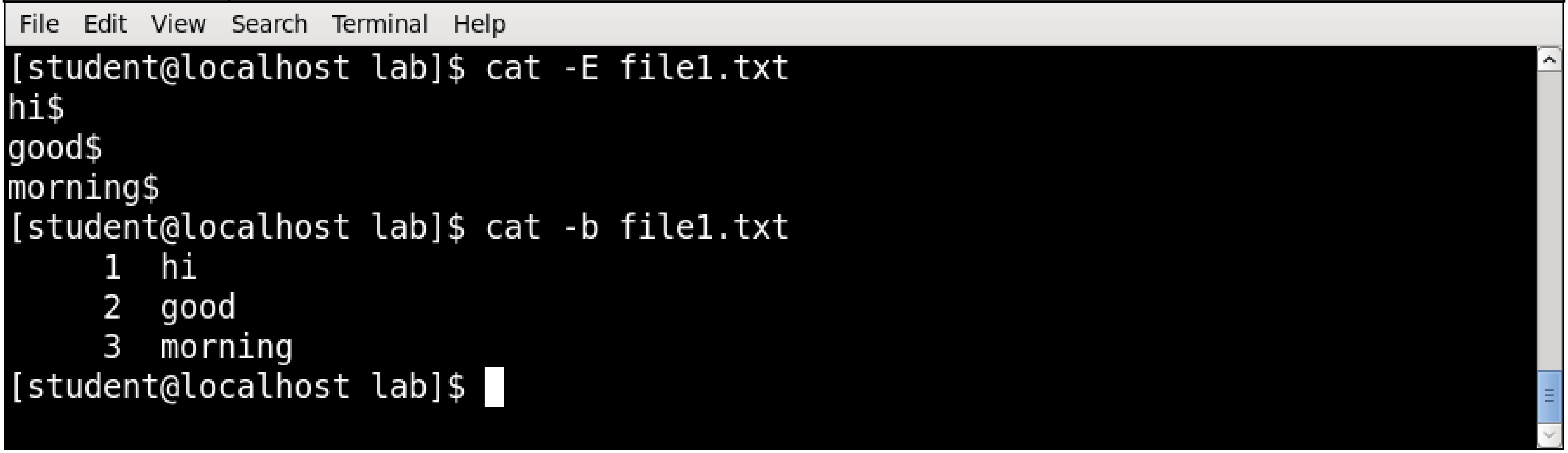
```
[student@localhost lab]$ cat > file1.txt
hi
good
morning
^C
[student@localhost lab]$ 
```



```
[student@localhost lab]$ cat file1.txt
hi
good
morning
[student@localhost lab]$ 
```

cat Command Example

cat -b	**Omits line numbers for blank space in the output**
cat -E	**Displays a $ (dollar sign) at the end of each line**

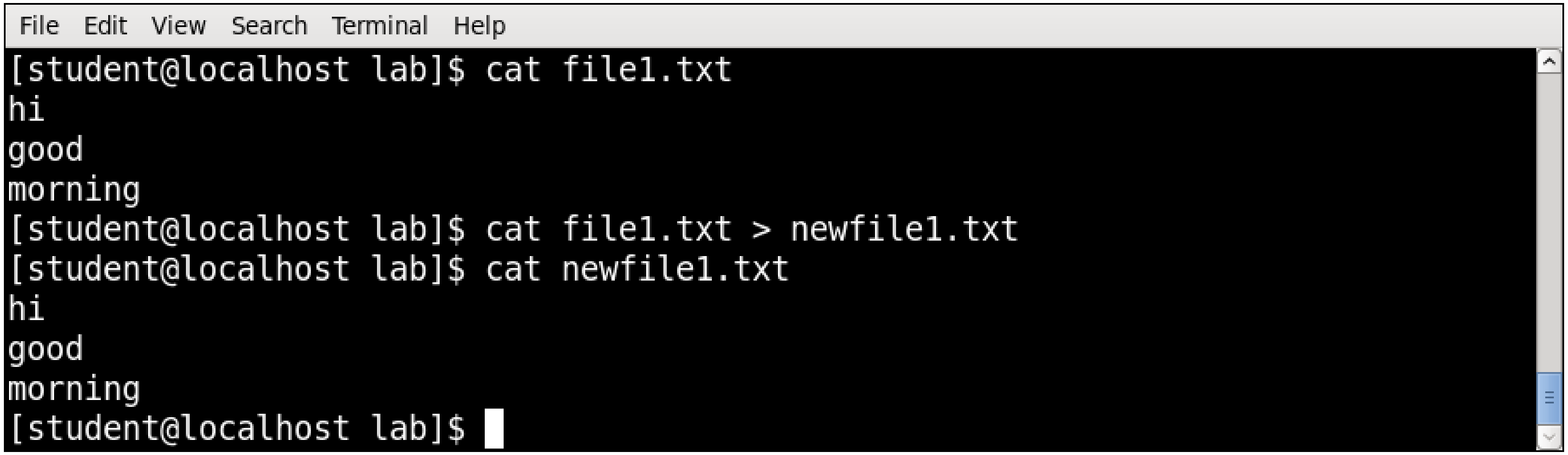


```
[student@localhost lab]$ cat -E file1.txt
hi$
good$
morning$
[student@localhost lab]$ cat -b file1.txt
     1	hi
     2	good
     3	morning
[student@localhost lab]$ 
```

cat Command Example

- **$cat file.txt > newfile.txt**

- Read the contents of file.txt and write them to newfile.txt, overwriting anything newfile.txt previously contained.
- If newfile.txt does not exist, it will be created.

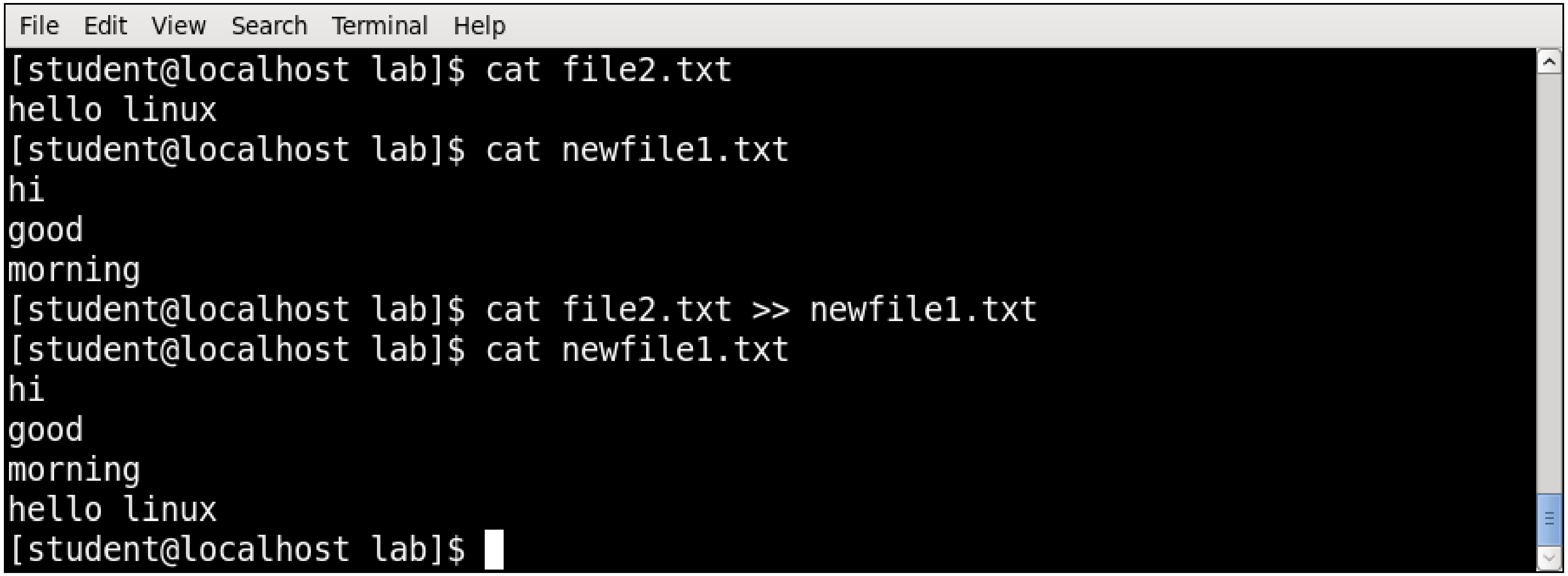


```
[student@localhost lab]$ cat file1.txt
hi
good
morning
[student@localhost lab]$ cat file1.txt > newfile1.txt
[student@localhost lab]$ cat newfile1.txt
hi
good
morning
[student@localhost lab]$ 
```

cat Command Example

- **$cat file.txt >> newfile.txt**

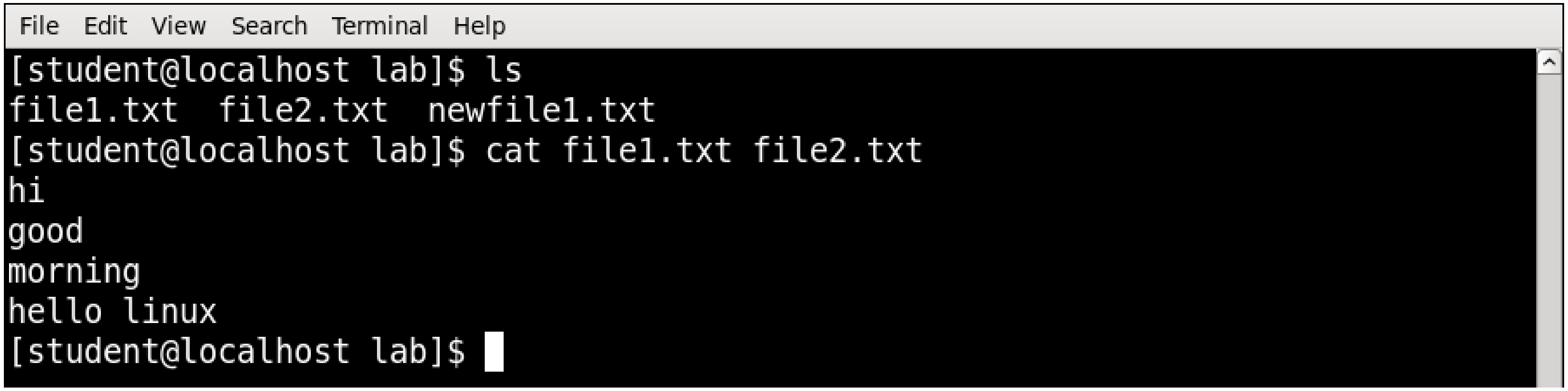
- Read the contents of **file.txt** and append them to the end of **newfile.txt**. If **newfile.txt** does not exist, it will be created.



```
[student@localhost lab]$ cat file2.txt
hello linux
[student@localhost lab]$ cat newfile1.txt
hi
good
morning
[student@localhost lab]$ cat file2.txt >> newfile1.txt
[student@localhost lab]$ cat newfile1.txt
hi
good
morning
hello linux
[student@localhost lab]$
```

cat Command Example

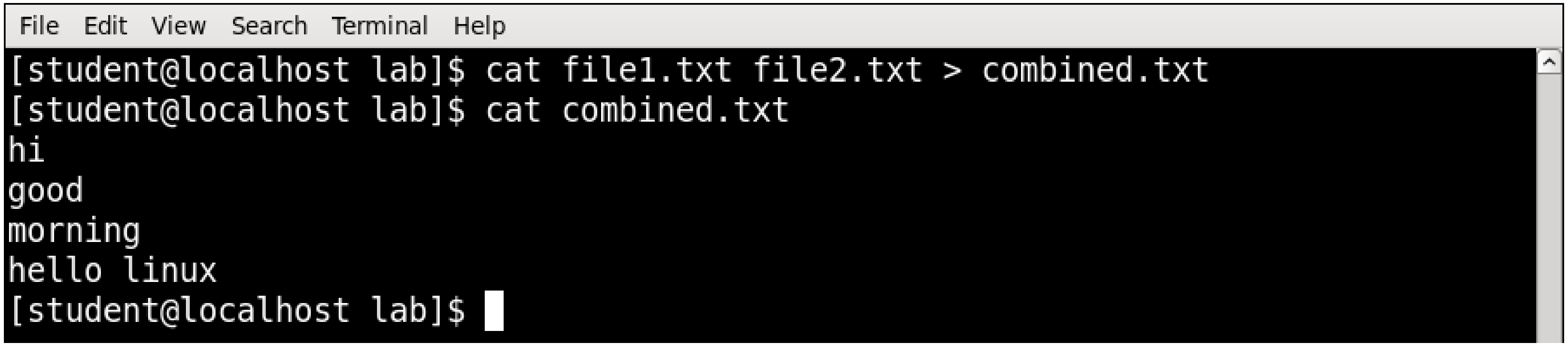
- **<u>cat file1.txt file2.txt</u>**
- It will read the contents of file1.txt and file2.txt and display the result in the terminal.



```
[student@localhost lab]$ ls
file1.txt  file2.txt  newfile1.txt
[student@localhost lab]$ cat file1.txt file2.txt
hi
good
morning
hello linux
[student@localhost lab]$ 
```

cat Command Example

- **<u>cat file1.txt file2.txt > combinedfile.txt</u>**
- It will concatenate the contents of file1.txt and file2.txt and write them to a new file combinedfile.txt using the (>) operator.
- If the combinedfile.txt file doesn’t exist the command will create it otherwise it will overwrite the file.



```
[student@localhost lab]$ cat file1.txt file2.txt > combined.txt
[student@localhost lab]$ cat combined.txt
hi
good
morning
hello linux
[student@localhost lab]$ 
```

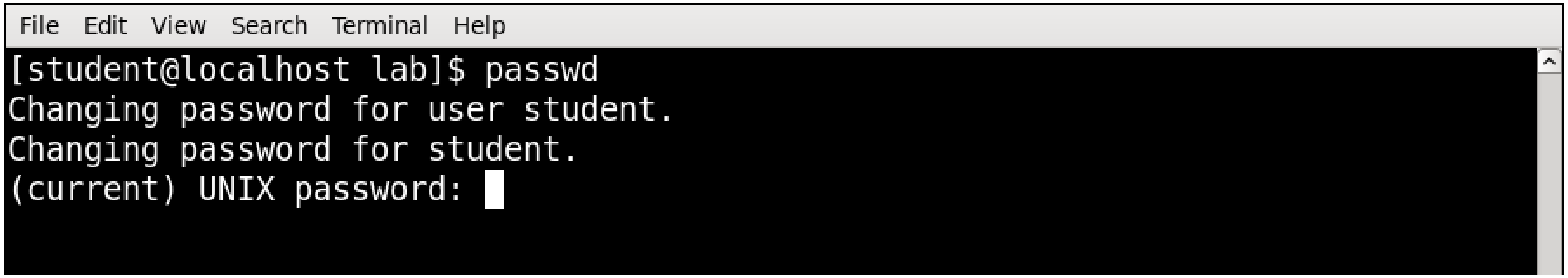
pwd (Print working directory) Command

- It prints the current working directory name with the complete path starting from root (/).

- **Syntax :**

  pwd [-OPTION]

- **Example :**



```
[student@localhost lab]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password: 
```

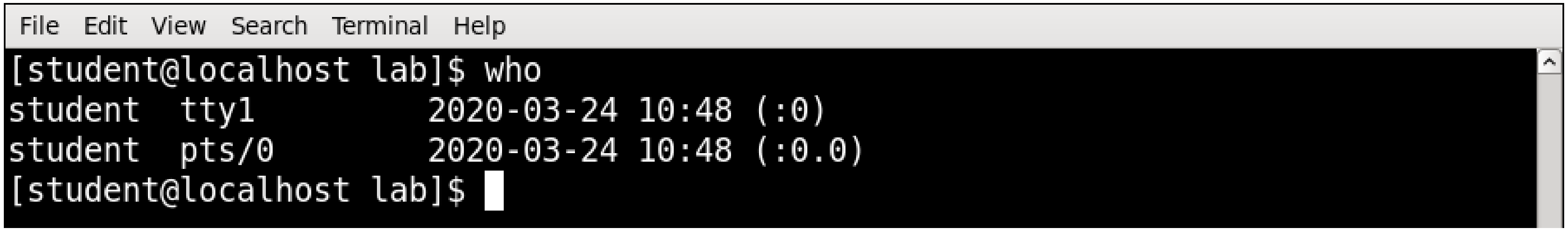
who Command

- It display the users that are currently logged into your Unix computer system.
- **Syntax :**

  who [-options] [filename]
- **Example :**

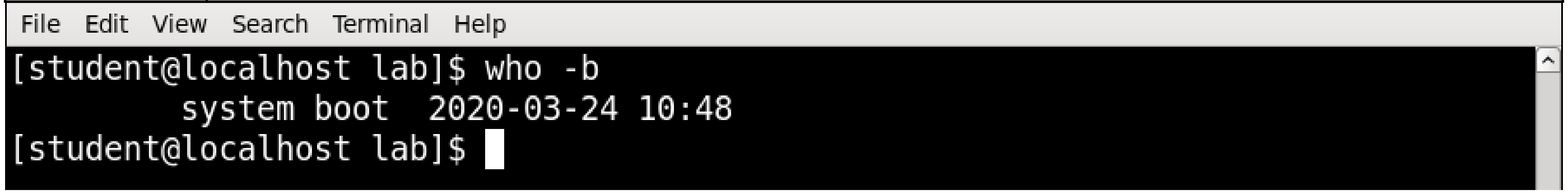
Option	Use
who -b	Display the time of the last system boot
who -H	Print a line of column headings
who -q	Displays all login names, and a count of all logged-on users
who -a	Display all details of current logged in user

who Command Example



```
[student@localhost lab]$ who
student  tty1         2020-03-24 10:48 (:0)
student  pts/0        2020-03-24 10:48 (:0.0)
[student@localhost lab]$ 
```

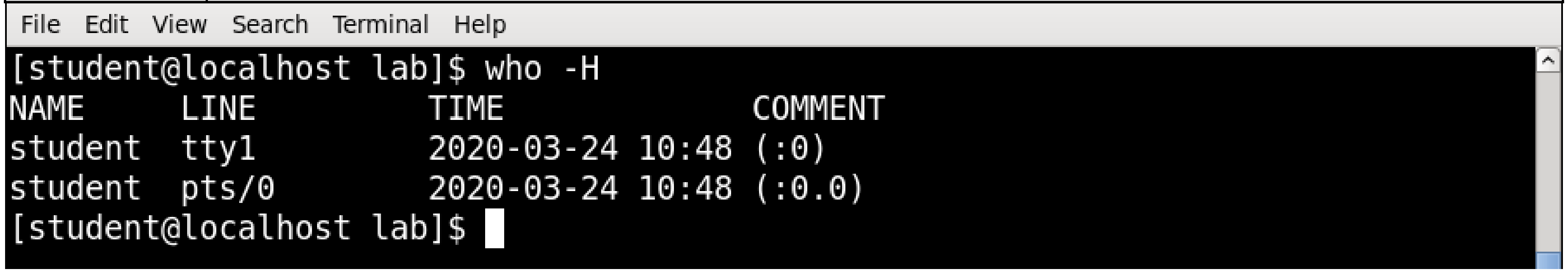
who -b	**Display the time of the last system boot**



```
[student@localhost lab]$ who -b
         system boot  2020-03-24 10:48
[student@localhost lab]$ 
```

who Command Example

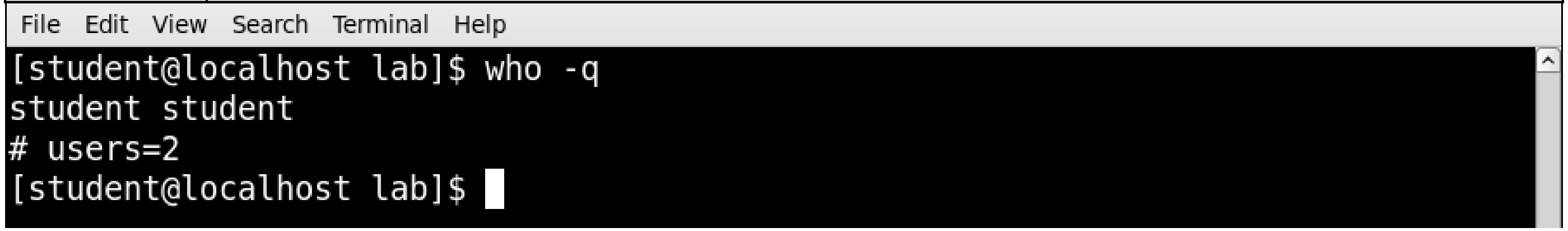
who -H	**Print a line of column headings**



```
[student@localhost lab]$ who -H
NAME     LINE         TIME             COMMENT
student  tty1         2020-03-24 10:48 (:0)
student  pts/0        2020-03-24 10:48 (:0.0)
[student@localhost lab]$ 
```

who Command Example

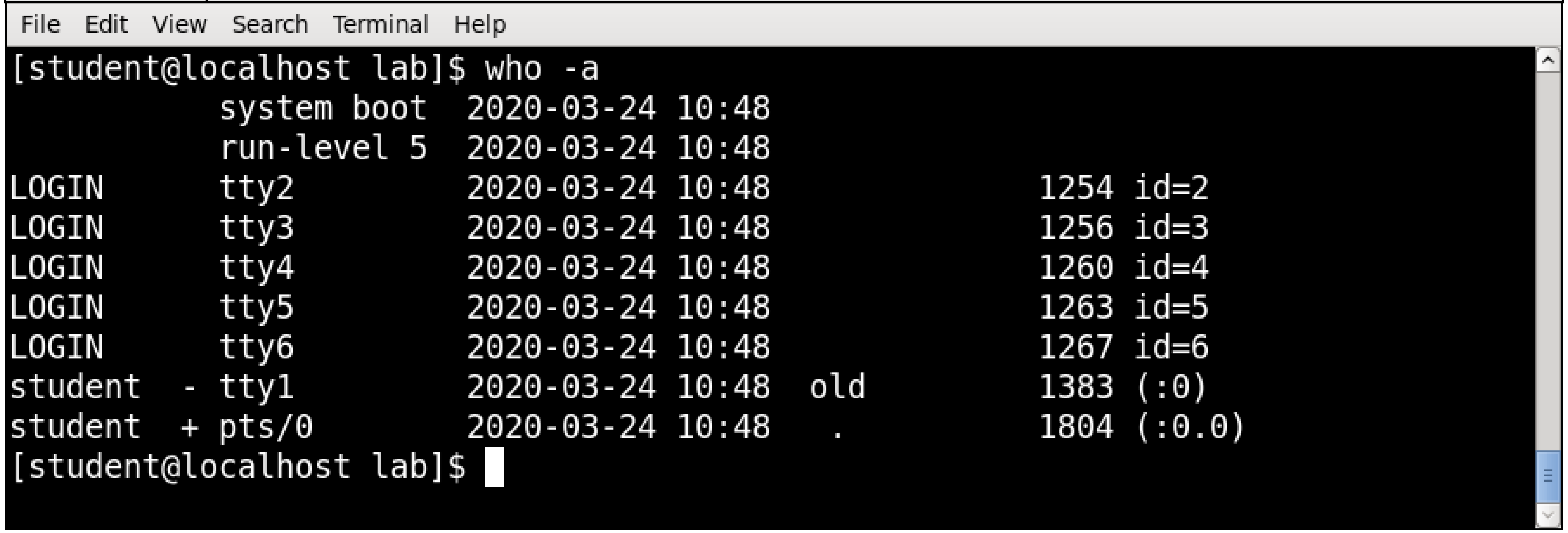
who -q	**Displays all login names, and a count of all logged-on users**



```
[student@localhost lab]$ who -q
student student
# users=2
[student@localhost lab]$ 
```

who Command Example

who -a	**Display all details of current logged in user**



```
[student@localhost lab]$ who -a
           system boot  2020-03-24 10:48
           run-level 5  2020-03-24 10:48
LOGIN      tty2         2020-03-24 10:48                 1254 id=2
LOGIN      tty3         2020-03-24 10:48                 1256 id=3
LOGIN      tty4         2020-03-24 10:48                 1260 id=4
LOGIN      tty5         2020-03-24 10:48                 1263 id=5
LOGIN      tty6         2020-03-24 10:48                 1267 id=6
student  - tty1         2020-03-24 10:48  old            1383 (:0)
student  + pts/0        2020-03-24 10:48   .             1804 (:0.0)
[student@localhost lab]$ 
```

whoami Command

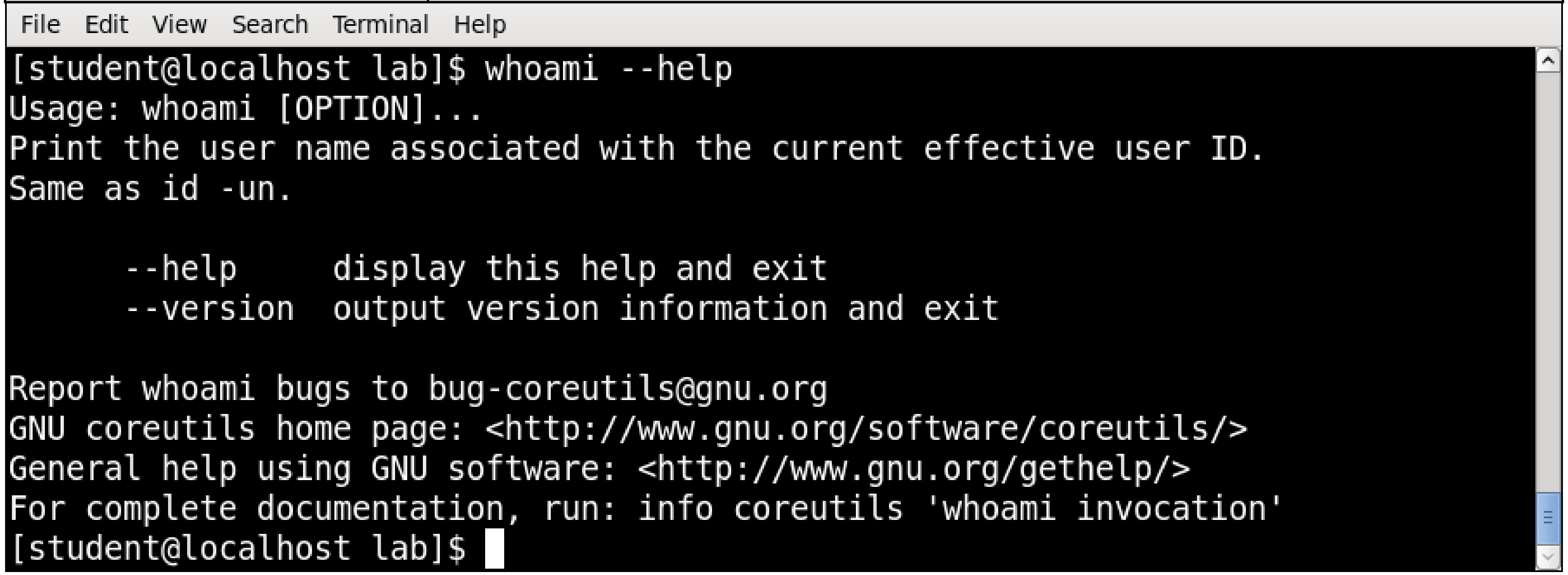
- This command prints the username associated with the current effective user ID.
- **Syntax :**

  whoami [-OPTION]
- **Example :**

Option	Use
whoami --help	Display a help message, and exit
whoami --version	Display version information, and exit

whoami Command Example

whoami --help	**Display a help message, and exit**



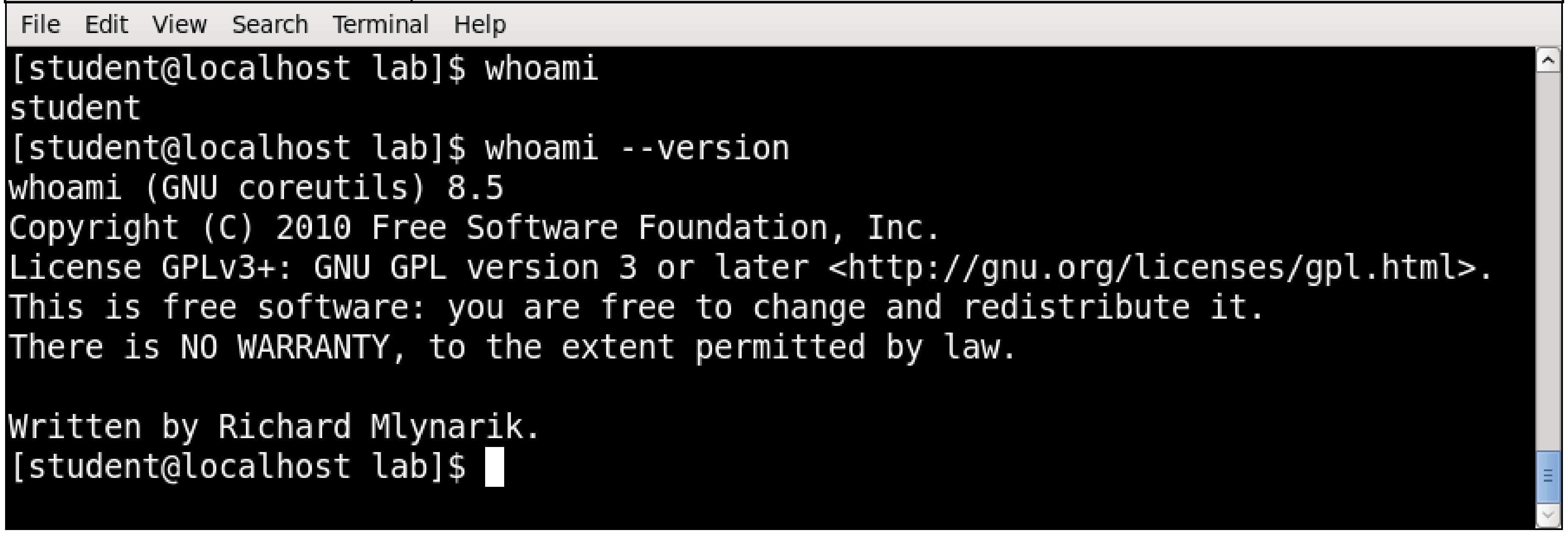
```
[student@localhost lab]$ whoami --help
Usage: whoami [OPTION]...
Print the user name associated with the current effective user ID.
Same as id -un.

      --help     display this help and exit
      --version  output version information and exit

Report whoami bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'whoami invocation'
[student@localhost lab]$
```

whoami Command Example

whoami --version	**Display version information, and exit**



```
[student@localhost lab]$ whoami
student
[student@localhost lab]$ whoami --version
whoami (GNU coreutils) 8.5
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Richard Mlynarik.
[student@localhost lab]$ 
```

uname (unix name) Command

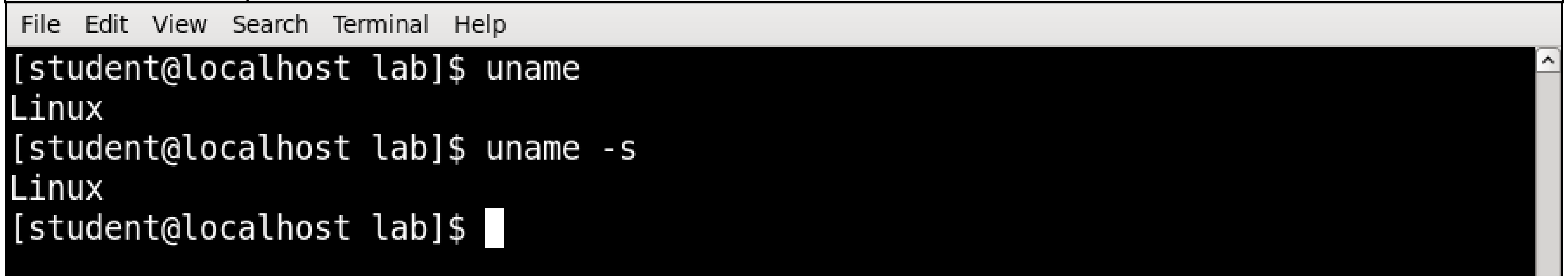
- Print information about the current system.
- **Syntax :**

  uname [-OPTION]
- **Example :**

Option	Use
uname -s	Print the kernel name
uname -n	Print the network node hostname
uname -v	Print the kernel version
uname -m	Print the machine hardware name
uname -o	Print the operating system

uname Command Example

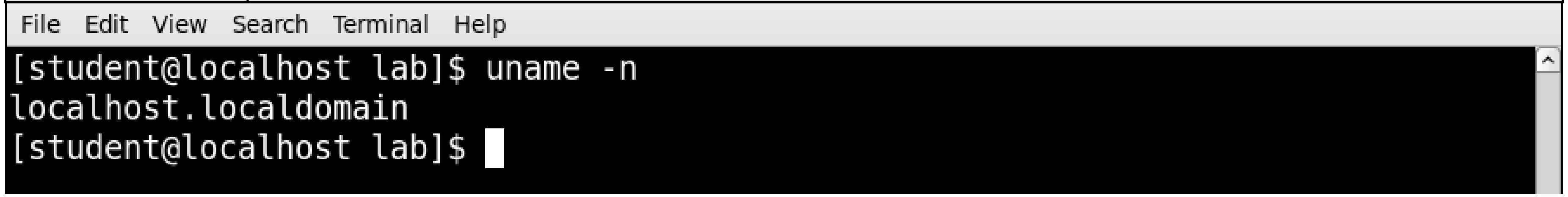
uname -s	**Print the kernel name**



```
[student@localhost lab]$ uname
Linux
[student@localhost lab]$ uname -s
Linux
[student@localhost lab]$ 
```

uname Command Example

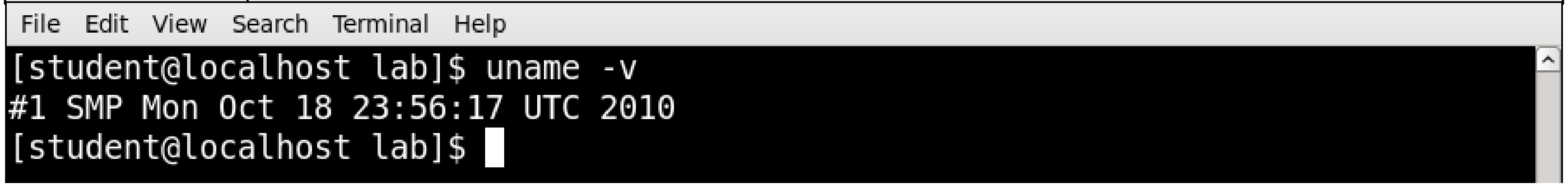
uname -n	**Print the network node hostname**



```
[student@localhost lab]$ uname -n
localhost.localdomain
[student@localhost lab]$ 
```

uname Command Example

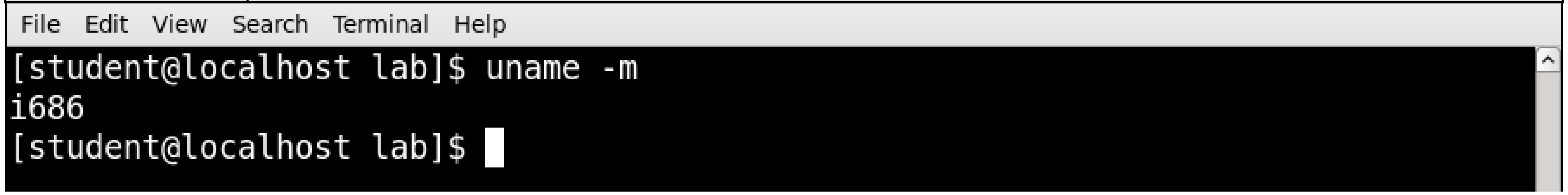
uname -v	**Print the kernel version**



```
[student@localhost lab]$ uname -v
#1 SMP Mon Oct 18 23:56:17 UTC 2010
[student@localhost lab]$ 
```

uname Command Example

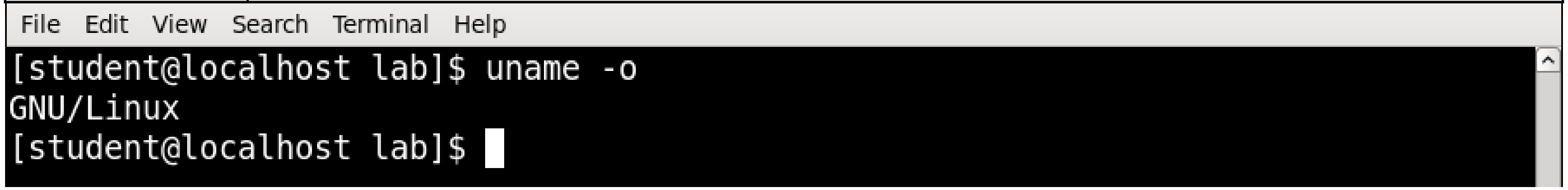
uname -m	**Print the machine hardware name**



```
[student@localhost lab]$ uname -m
i686
[student@localhost lab]$ 
```

uname Command Example

uname -o	**Print the operating system**



```
[student@localhost lab]$ uname -o
GNU/Linux
[student@localhost lab]$ 
```

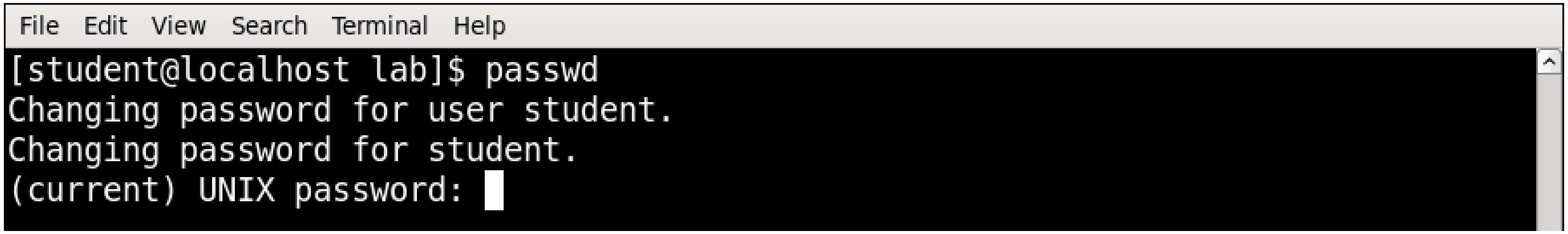
passwd Command

- The passwd command is used to change the password of a user account.

- **Syntax :**

  passwd [-options] [username]

- **Example :**



File Edit View Search Terminal Help

```
[student@localhost lab]$ passwd
Changing password for user student.
Changing password for student.
(current) UNIX password: 
```

mkdir Command

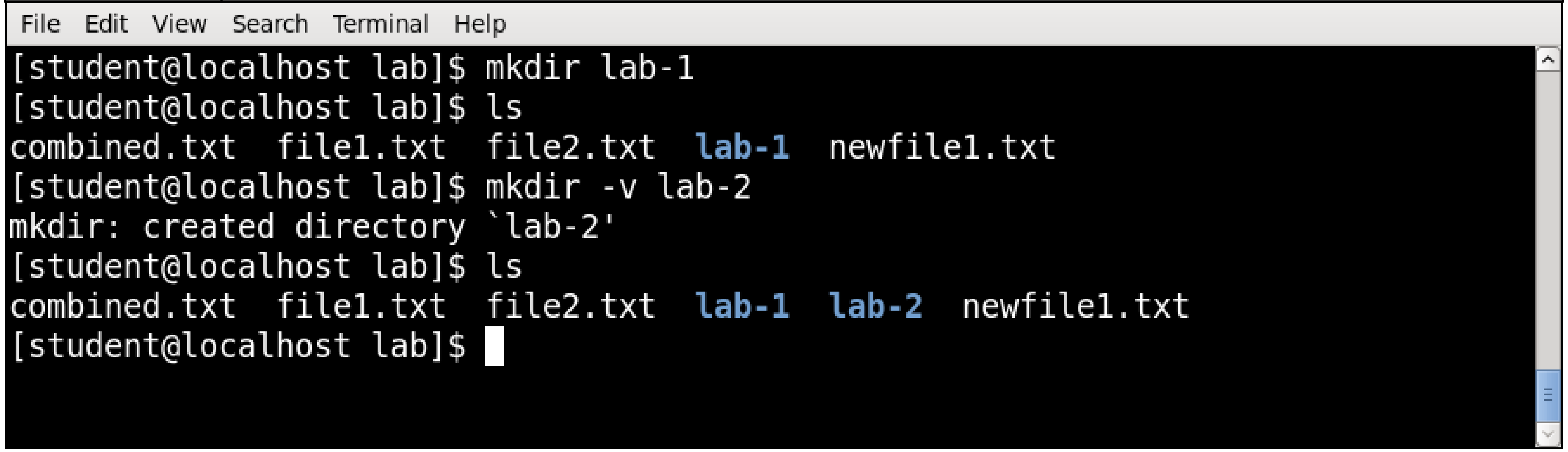
- This command is used to make Directories.
- **Syntax :**

  mkdir [-OPTION] DIRECTORY
- **Example :**

Option	Use
mkdir -v	Print a message for each created directory
mkdir -p	No error if existing, make parent directories as needed
mkdir -m	To control the permissions of new directories

mkdir Command Example

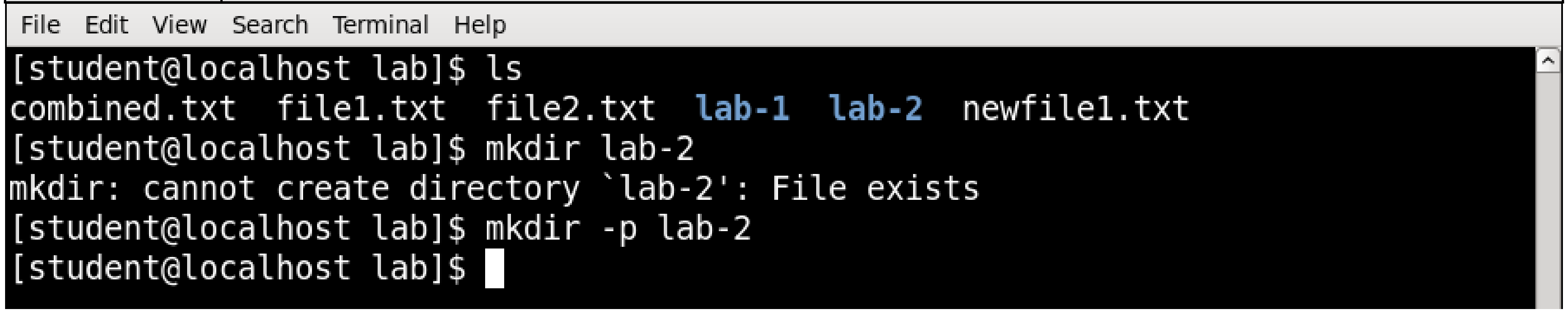
mkdir -v	**Print a message for each created directory**



```
[student@localhost lab]$ mkdir lab-1
[student@localhost lab]$ ls
combined.txt  file1.txt  file2.txt  lab-1  newfile1.txt
[student@localhost lab]$ mkdir -v lab-2
mkdir: created directory `lab-2'
[student@localhost lab]$ ls
combined.txt  file1.txt  file2.txt  lab-1  lab-2  newfile1.txt
[student@localhost lab]$ 
```

mkdir Command Example

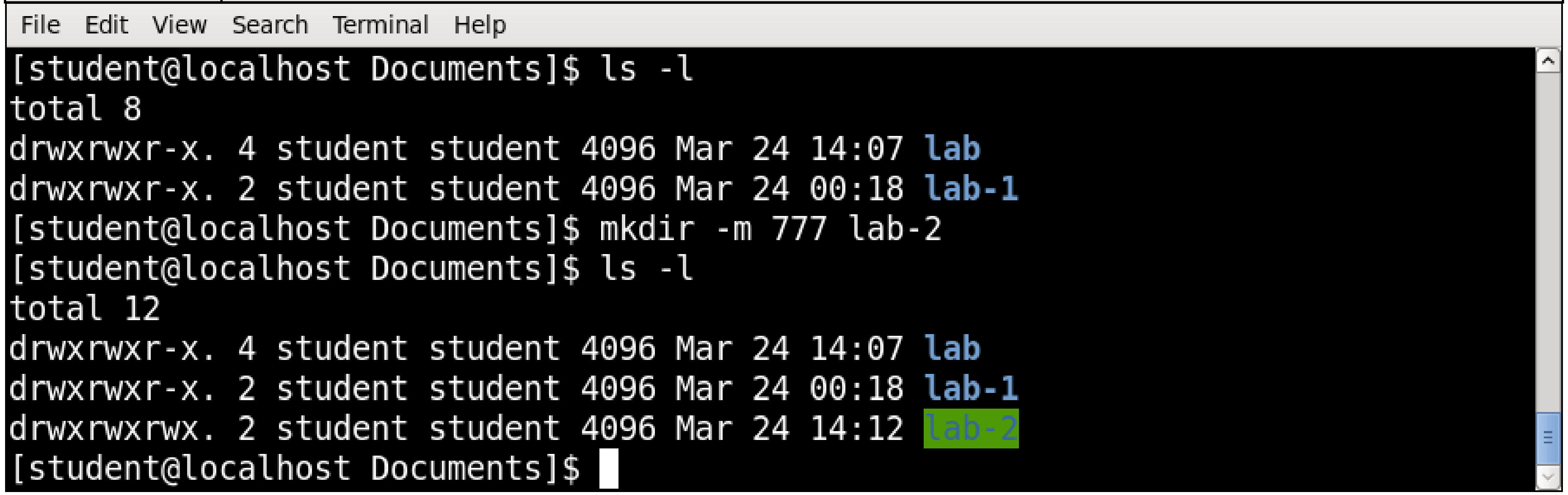
mkdir -p	**No error if existing, make parent directories as needed**



```
[student@localhost lab]$ ls
combined.txt  file1.txt  file2.txt  lab-1  lab-2  newfile1.txt
[student@localhost lab]$ mkdir lab-2
mkdir: cannot create directory `lab-2': File exists
[student@localhost lab]$ mkdir -p lab-2
[student@localhost lab]$ 
```

mkdir Command Example

mkdir -m	**To control the permissions of new directories**



```
[student@localhost Documents]$ ls -l
total 8
drwxrwxr-x. 4 student student 4096 Mar 24 14:07 lab
drwxrwxr-x. 2 student student 4096 Mar 24 00:18 lab-1
[student@localhost Documents]$ mkdir -m 777 lab-2
[student@localhost Documents]$ ls -l
total 12
drwxrwxr-x. 4 student student 4096 Mar 24 14:07 lab
drwxrwxr-x. 2 student student 4096 Mar 24 00:18 lab-1
drwxrwxrwx. 2 student student 4096 Mar 24 14:12 lab-2
[student@localhost Documents]$
```

rmdir Command

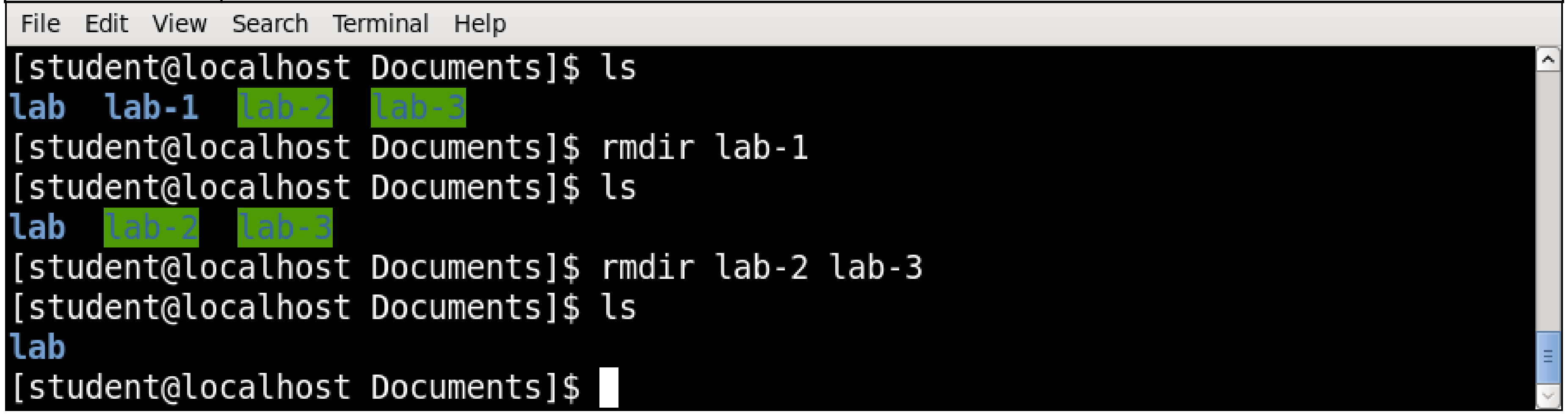
- This command removes empty directories from your filesystem.
- **Syntax :**

  rmdir [-OPTION] DIRECTORY
- **Example :**

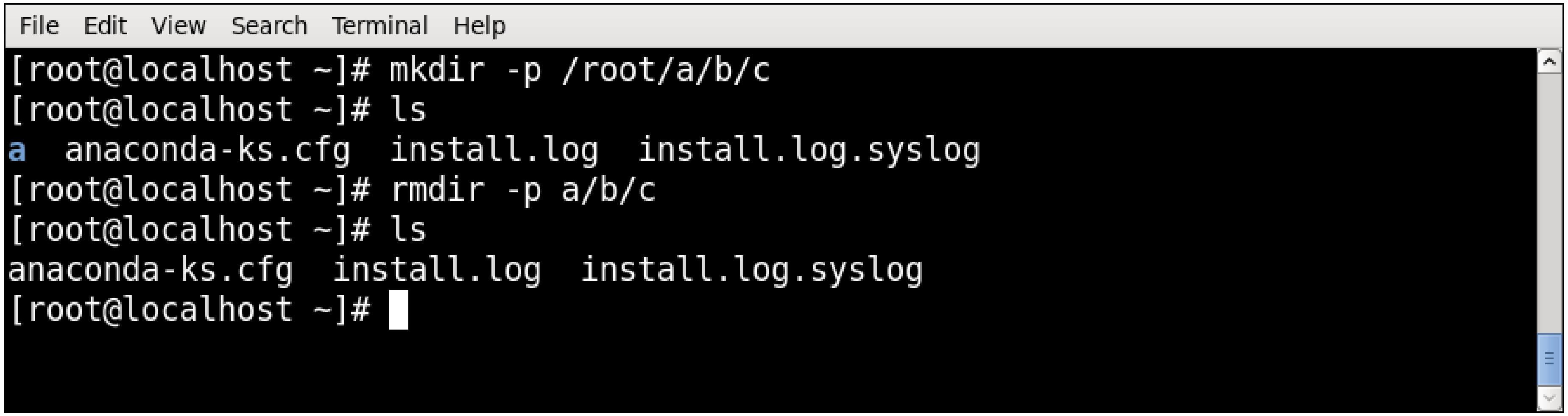
Option	Use
rmdir -p	Remove directory and its ancestors… e.g., ‘rmdir -p a/b/c’ is similar to ‘rmdir a/b/c a/b a’

rmdir Command Example

rmdir -p	**Remove directory and its ancestors…** **e.g., ‘rmdir -p a/b/c’ is similar to ‘rmdir a/b/c a/b a’**



```
[student@localhost Documents]$ ls
lab  lab-1  lab-2  lab-3
[student@localhost Documents]$ rmdir lab-1
[student@localhost Documents]$ ls
lab  lab-2  lab-3
[student@localhost Documents]$ rmdir lab-2 lab-3
[student@localhost Documents]$ ls
lab
[student@localhost Documents]$
```



```
[root@localhost ~]# mkdir -p /root/a/b/c
[root@localhost ~]# ls
a  anaconda-ks.cfg  install.log  install.log.syslog
[root@localhost ~]# rmdir -p a/b/c
[root@localhost ~]# ls
anaconda-ks.cfg  install.log  install.log.syslog
[root@localhost ~]#
```

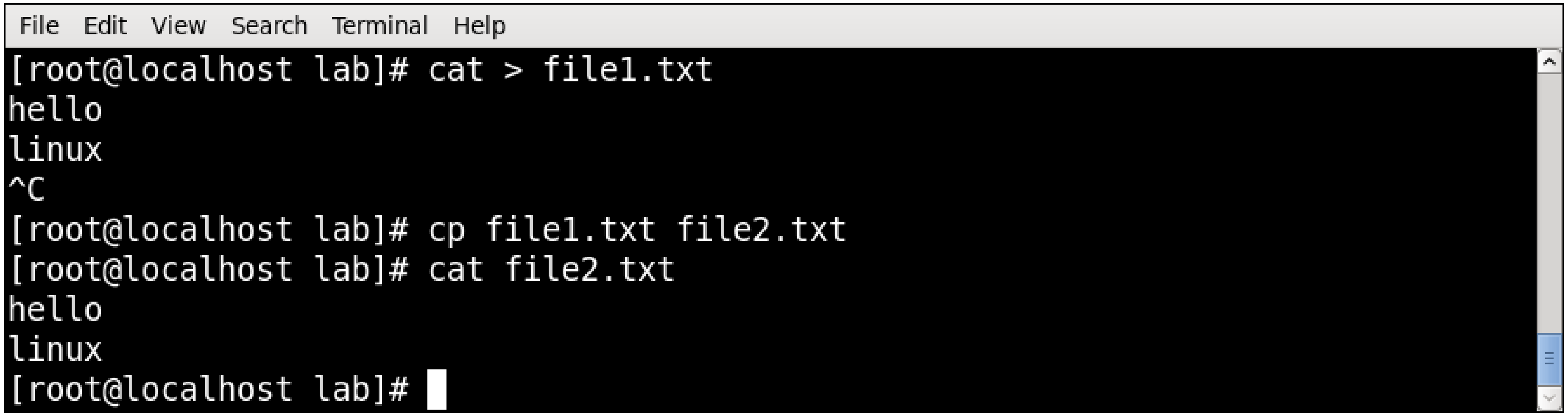
cp(copy) Command

- This command is used to copy files and directories.
- **Syntax :**

  cp [option] source destination/directory
- **Example :**

Option	Use
cp -i	Interactive - ask before overwrite
cp -f	Force copy by removing the destination file if needed
cp -n	Do not overwrite an existing file
cp -u	Update - copy when source is newer than destination
cp -s	Make symbolic links instead of copying
cp -R	Copy directories recursively
cp -v	Print informative messages

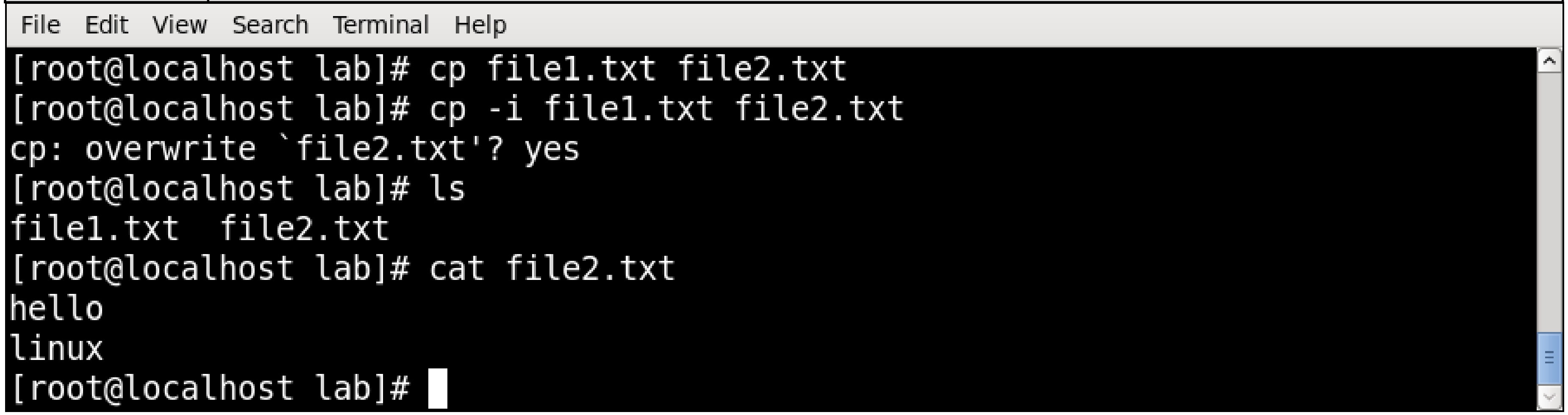
cp Command Example



```
[root@localhost lab]# cat > file1.txt
hello
linux
^C
[root@localhost lab]# cp file1.txt file2.txt
[root@localhost lab]# cat file2.txt
hello
linux
[root@localhost lab]# 
```

cp Command Example

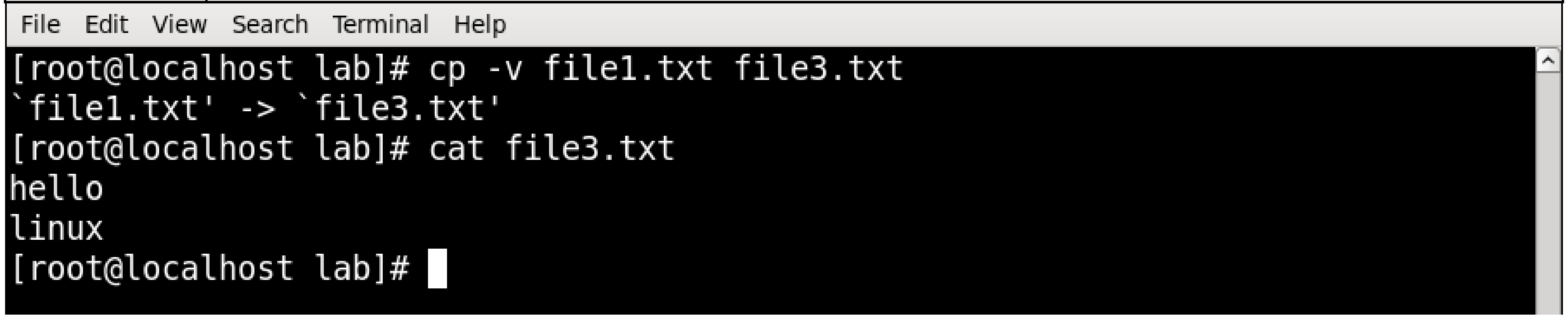
cp -i	**Interactive - ask before overwrite**



```
[root@localhost lab]# cp file1.txt file2.txt
[root@localhost lab]# cp -i file1.txt file2.txt
cp: overwrite `file2.txt'? yes
[root@localhost lab]# ls
file1.txt  file2.txt
[root@localhost lab]# cat file2.txt
hello
linux
[root@localhost lab]# 
```

cp Command Example

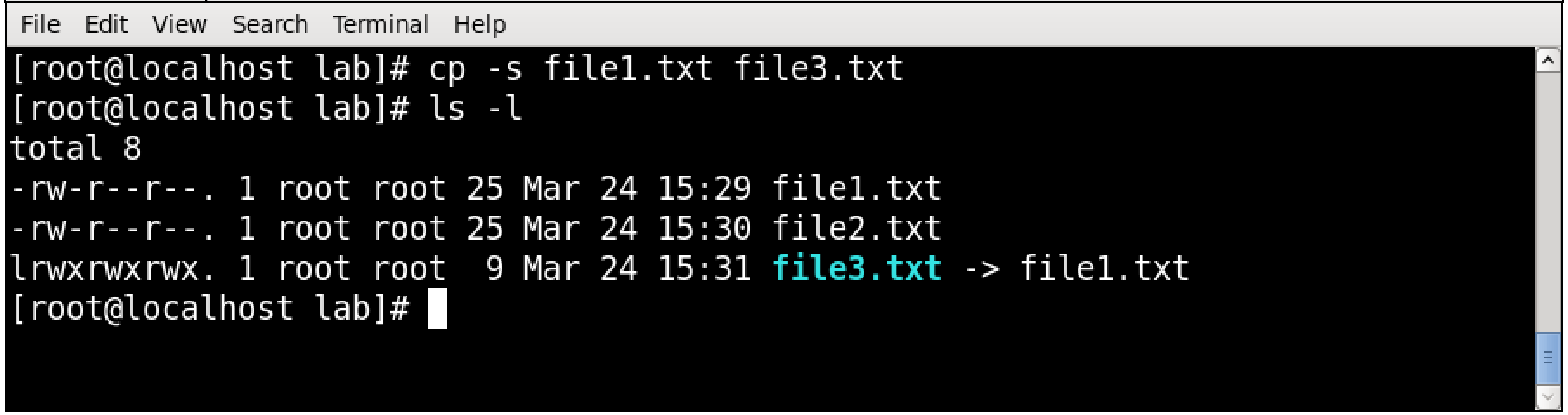
cp -v	**Print informative messages**



```
[root@localhost lab]# cp -v file1.txt file3.txt
`file1.txt' -> `file3.txt'
[root@localhost lab]# cat file3.txt
hello
linux
[root@localhost lab]# 
```

cp Command Example

cp -s	**Make symbolic links instead of copying**



```
[root@localhost lab]# cp -s file1.txt file3.txt
[root@localhost lab]# ls -l
total 8
-rw-r--r--. 1 root root 25 Mar 24 15:29 file1.txt
-rw-r--r--. 1 root root 25 Mar 24 15:30 file2.txt
lrwxrwxrwx. 1 root root  9 Mar 24 15:31 file3.txt -> file1.txt
[root@localhost lab]# 
```

mv(move) Command

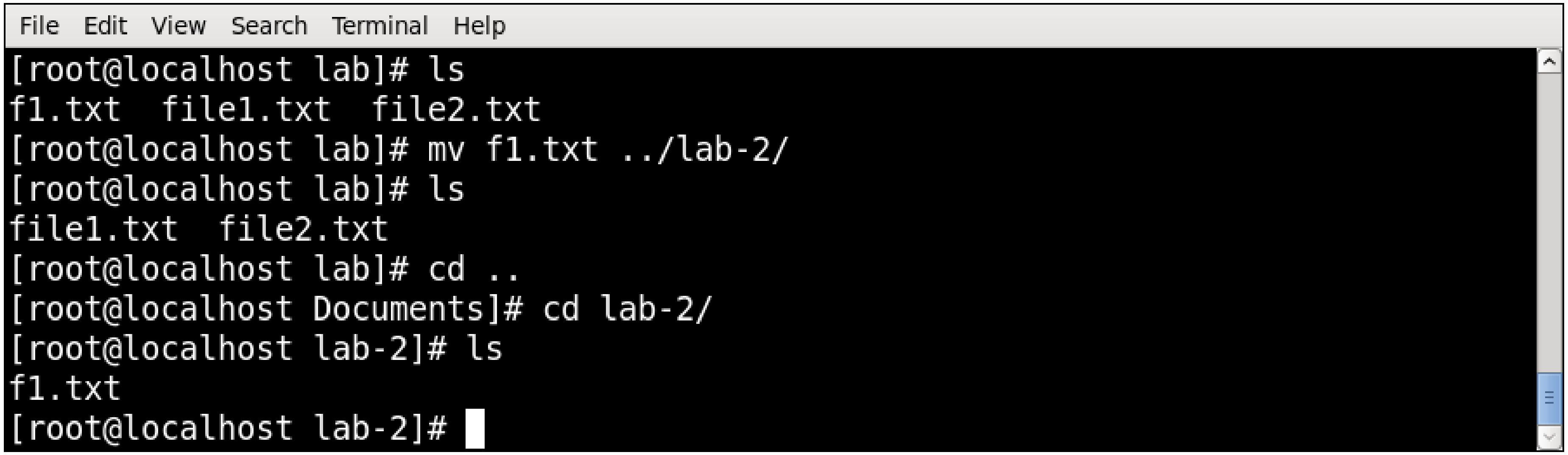
- mv command is used to move files and directories.
- **Syntax :**

  mv [-*options*] *source dest*

- **Example :**

Option	Use
mv -i	Interactive prompt before overwrite
mv -f	Force move by overwriting destination file without prompt
mv -n	Never overwrite any existing file
mv -u	Update - move when source is newer than destination
mv -v	Print informative messages

mv Command Example



```
[root@localhost lab]# ls
f1.txt  file1.txt  file2.txt
[root@localhost lab]# mv f1.txt ../lab-2/
[root@localhost lab]# ls
file1.txt  file2.txt
[root@localhost lab]# cd ..
[root@localhost Documents]# cd lab-2/
[root@localhost lab-2]# ls
f1.txt
[root@localhost lab-2]# 
```

rm(remove) Command

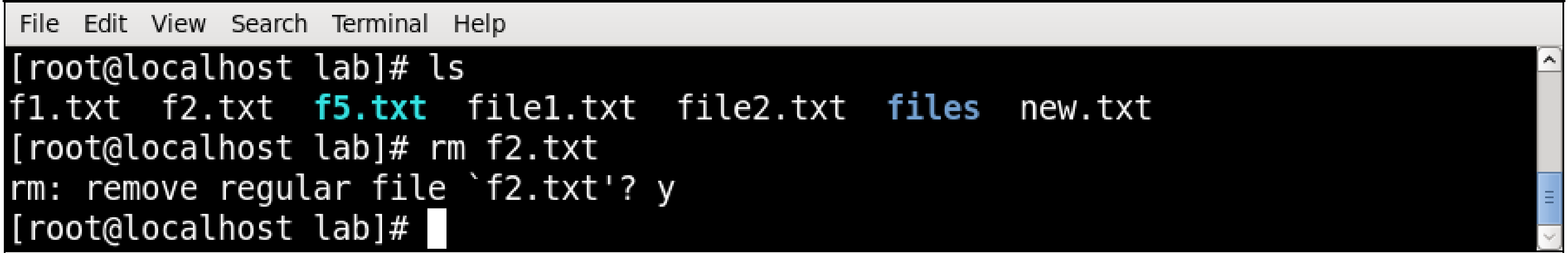
- The ‘**rm**’ command is used to delete files and directories.
- **Syntax :**

  rm [-OPTION] Filename
- **Example :**

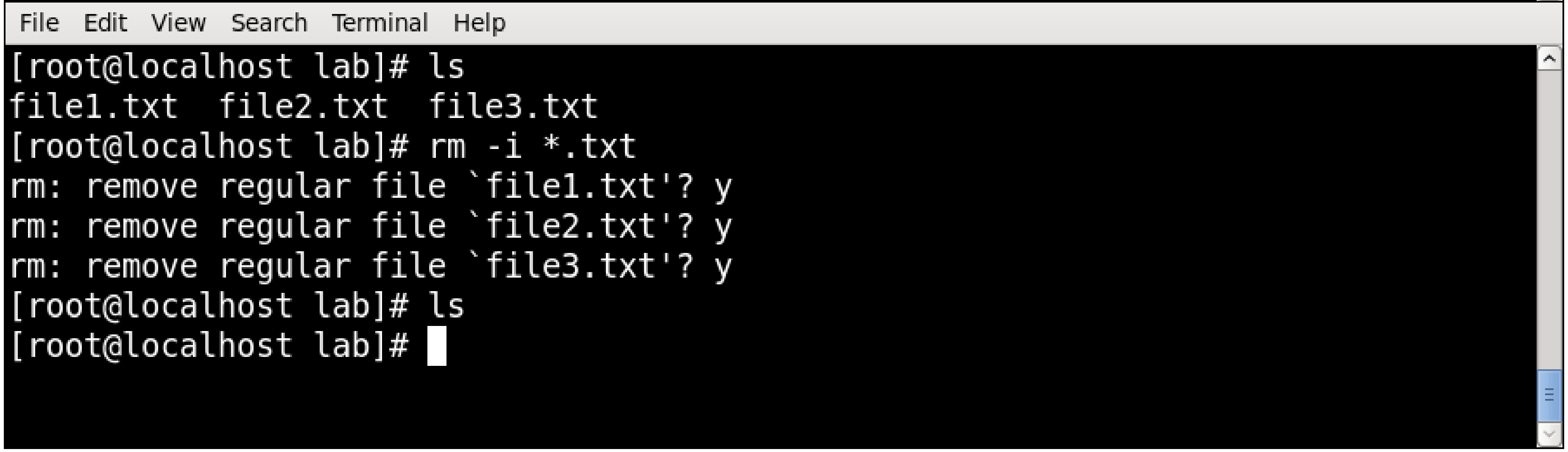
Option	Use
rm -i	Prompt before every removal
rm -d	Delete a empty directory
rm -r	Remove directories and their contents recursively
rm -f	To remove the file forcefully

rm Command Example

rm -i	**Prompt before every removal**



```
[root@localhost lab]# ls
f1.txt  f2.txt  f5.txt  file1.txt  file2.txt  files  new.txt
[root@localhost lab]# rm f2.txt
rm: remove regular file `f2.txt'? y
[root@localhost lab]#
```



```
[root@localhost lab]# ls
file1.txt  file2.txt  file3.txt
[root@localhost lab]# rm -i *.txt
rm: remove regular file `file1.txt'? y
rm: remove regular file `file2.txt'? y
rm: remove regular file `file3.txt'? y
[root@localhost lab]# ls
[root@localhost lab]#
```

cut Command

- The cut command extracts a given number of characters or columns from a file.

- **Syntax :**

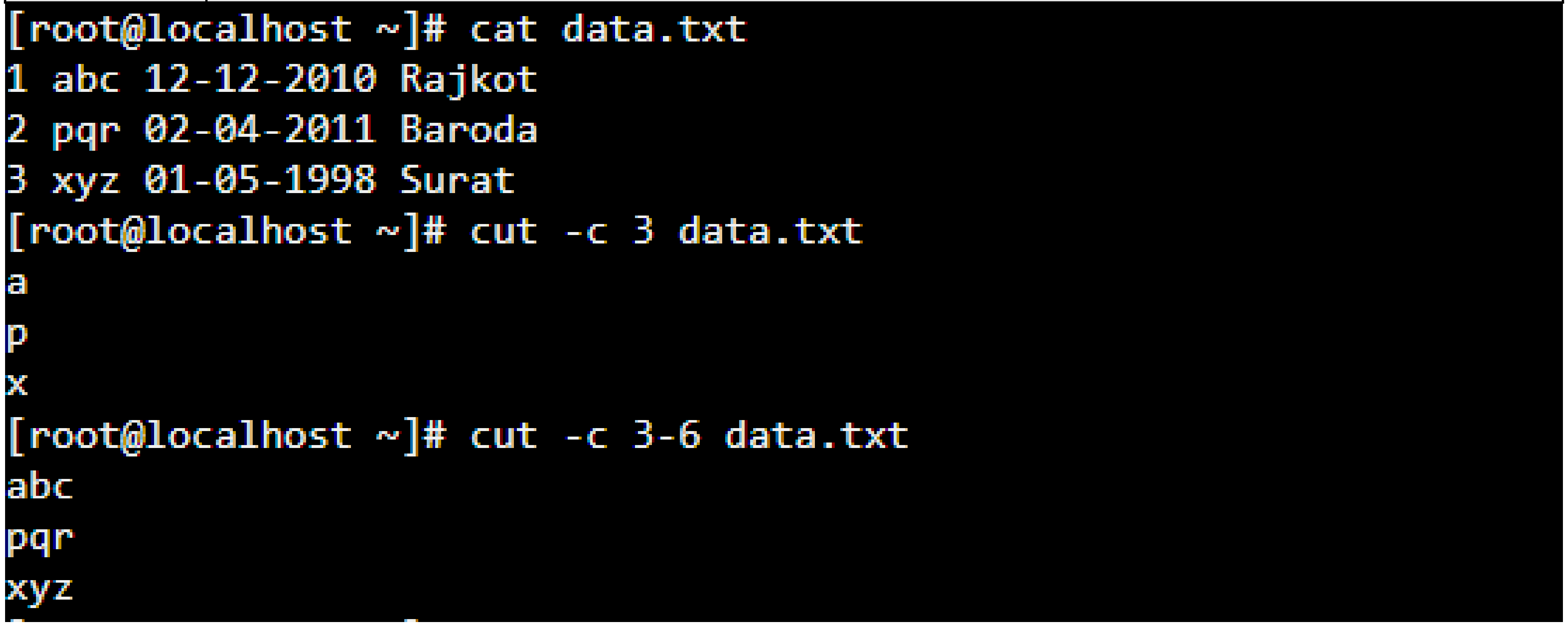
  cut [-options] [file]

- **Example :**

Option	Use
cut -c	Select only the characters from each line as specified in LIST
cut -b	Select only the bytes from each line as specified in LIST
cut -f	Cuts the input file using list of field. The default field to be used TAB. The default behavior can be overwritten by use of -d option
cut -d	Specifies a delimiter to by used as a field. Default field is TAB and this option overwrites this default behavior

cut Command Example

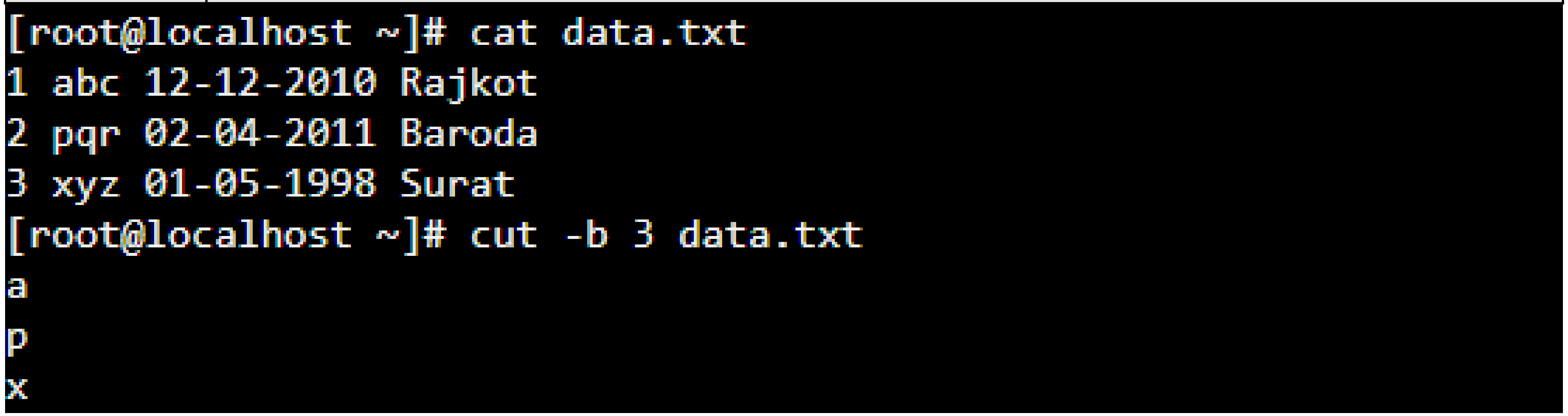
cut -c	**Select only the characters from each line as specified in LIST**



```
[root@localhost ~]# cat data.txt
1 abc 12-12-2010 Rajkot
2 pqr 02-04-2011 Baroda
3 xyz 01-05-1998 Surat
[root@localhost ~]# cut -c 3 data.txt
a
p
x
[root@localhost ~]# cut -c 3-6 data.txt
abc
pqr
xyz
```

cut Command Example

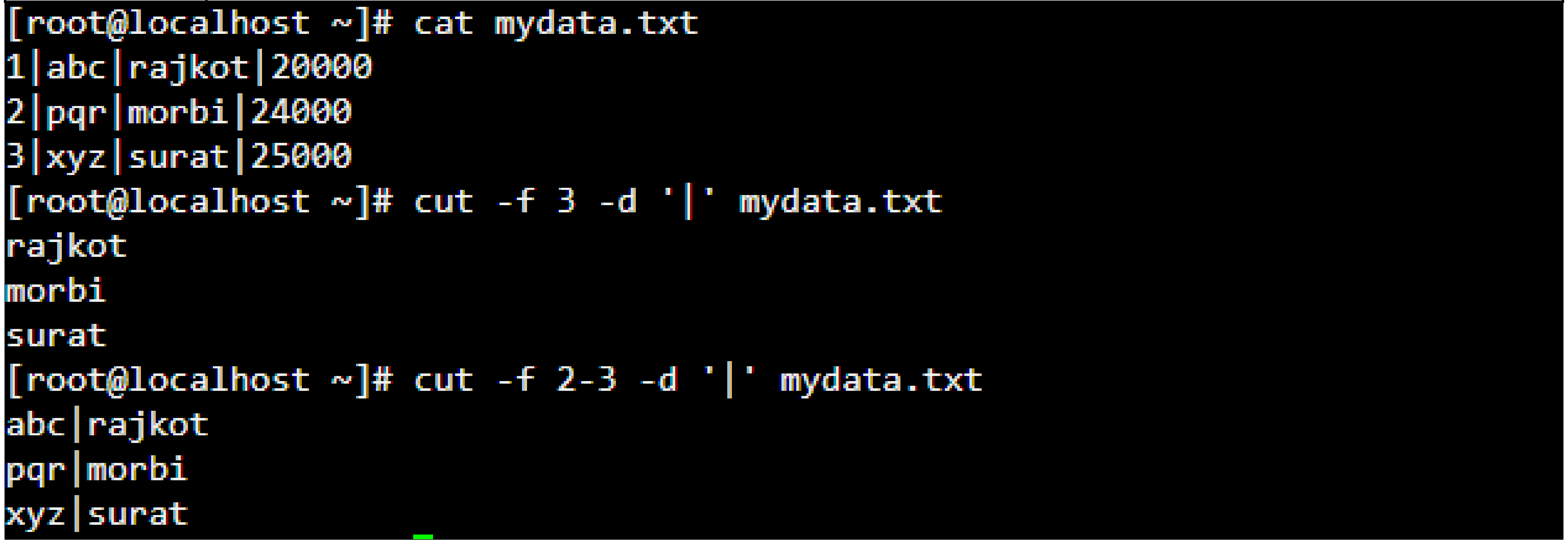
cut -b	**Select only the bytes from each line as specified in LIST**



```
[root@localhost ~]# cat data.txt
1 abc 12-12-2010 Rajkot
2 pqr 02-04-2011 Baroda
3 xyz 01-05-1998 Surat
[root@localhost ~]# cut -b 3 data.txt
a
p
x
```

cut Command Example

cut -f	Cuts the input file using list of field. The default field to be used TAB. The default behavior can be overwritten by use of -d option
cut -d	Specifies a delimiter to by used as a field. Default field is TAB and this option overwrites this default behavior



```
[root@localhost ~]# cat mydata.txt
1|abc|rajkot|20000
2|pqr|morbi|24000
3|xyz|surat|25000
[root@localhost ~]# cut -f 3 -d '|' mydata.txt
rajkot
morbi
surat
[root@localhost ~]# cut -f 2-3 -d '|' mydata.txt
abc|rajkot
pqr|morbi
xyz|surat
```

paste Command

- The paste command displays the corresponding lines of multiple files side-by-side.

- **Syntax :**

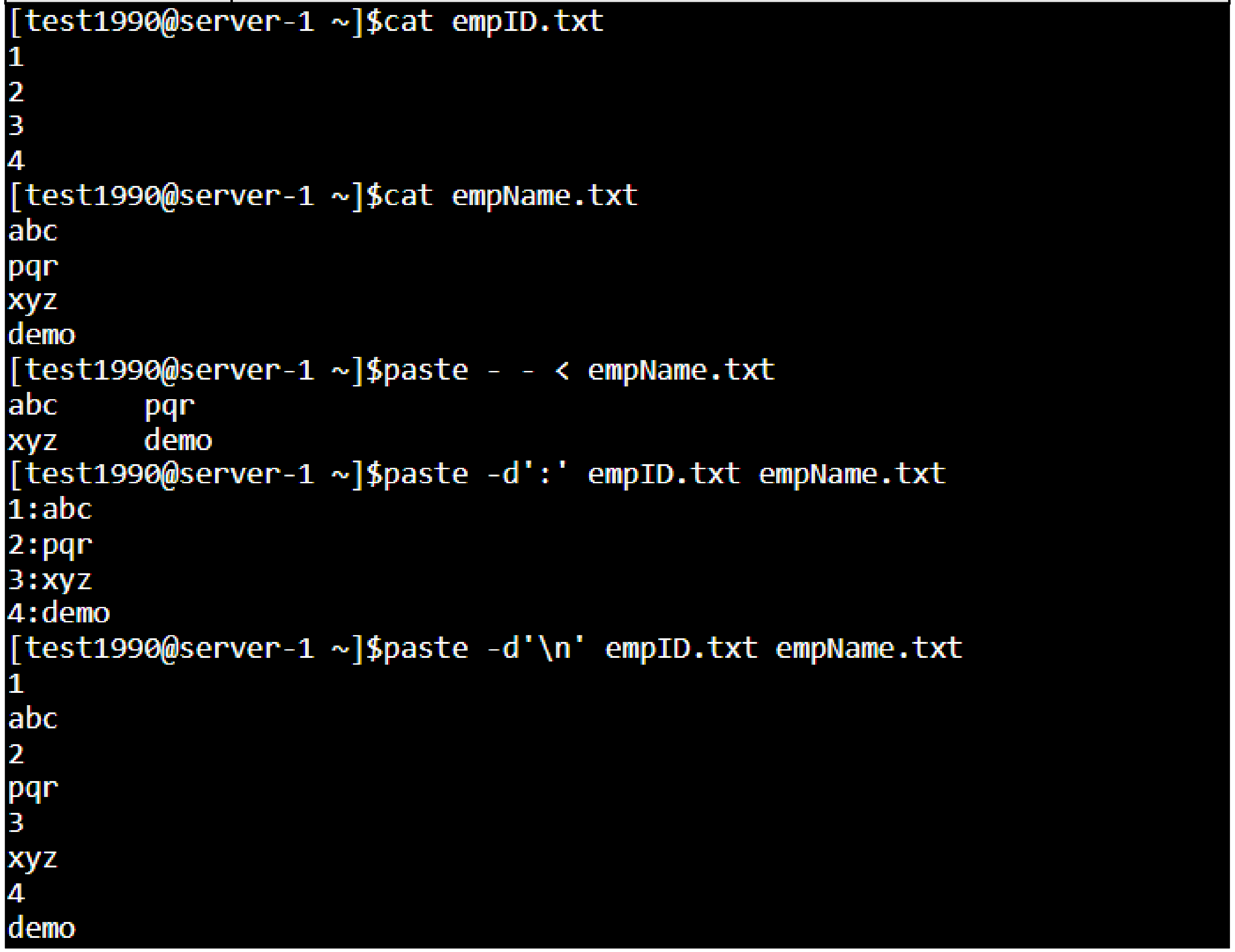
  paste [-options] [file]

- **Example :**

Option	Use
paste -d	Reuse characters from LIST instead of tabs
paste -s	Paste one file at a time instead of in parallel

paste Command Example

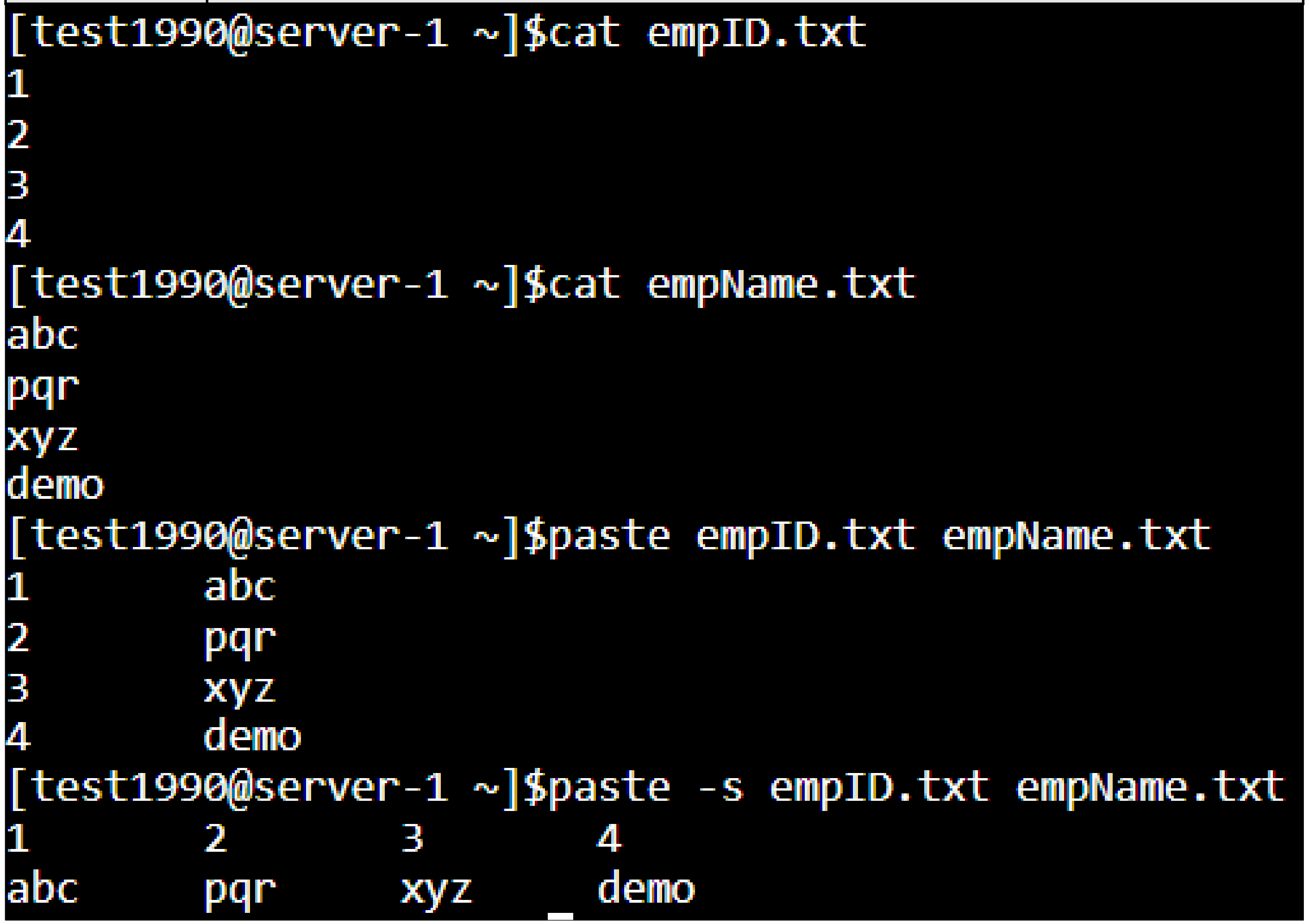
paste -d	**Reuse characters from LIST instead of tabs**



```
[test1990@server-1 ~]$cat empID.txt
1
2
3
4
[test1990@server-1 ~]$cat empName.txt
abc
pqr
xyz
demo
[test1990@server-1 ~]$paste - - < empName.txt
abc     pqr
xyz     demo
[test1990@server-1 ~]$paste -d':' empID.txt empName.txt
1:abc
2:pqr
3:xyz
4:demo
[test1990@server-1 ~]$paste -d'\n' empID.txt empName.txt
1
abc
2
pqr
3
xyz
4
demo
```

paste Command Example

paste -s	**Paste one file at a time instead of in parallel**



```
[test1990@server-1 ~]$cat empID.txt
1
2
3
4
[test1990@server-1 ~]$cat empName.txt
abc
pqr
xyz
demo
[test1990@server-1 ~]$paste empID.txt empName.txt
1       abc
2       pqr
3       xyz
4       demo
[test1990@server-1 ~]$paste -s empID.txt empName.txt
1       2       3       4
abc     pqr     xyz     demo
```

more Command

- The more command is a command line utility for viewing the contents of a file or files once screen at a time.

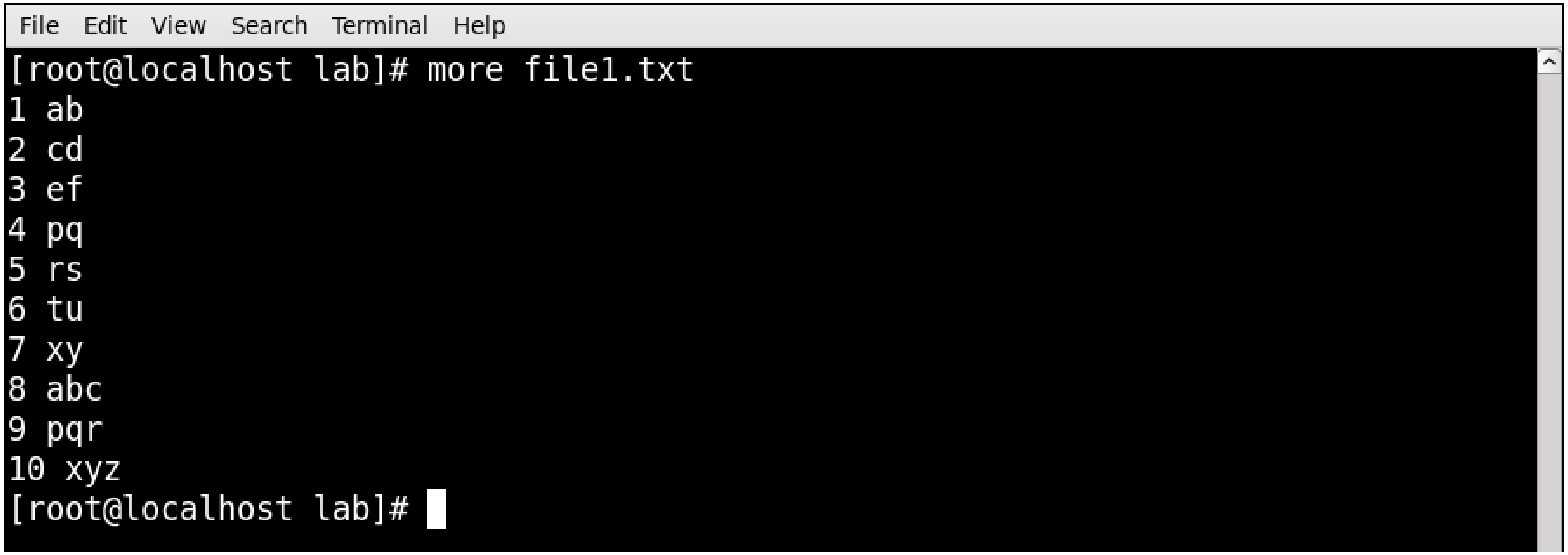
- **Syntax :**

  more [-options] [file]

- **Example :**

Option	Use
more -c	Clear screen before displaying
more -number	To Specify how many lines are printed in the screen for a given file
more -s	Doesn’t display extra blank lines

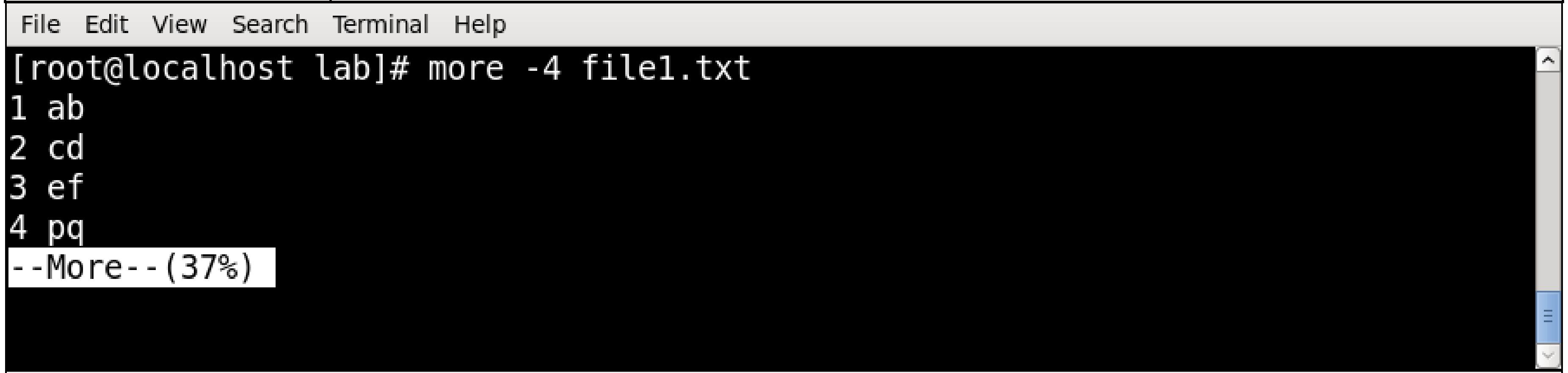
more Command Example



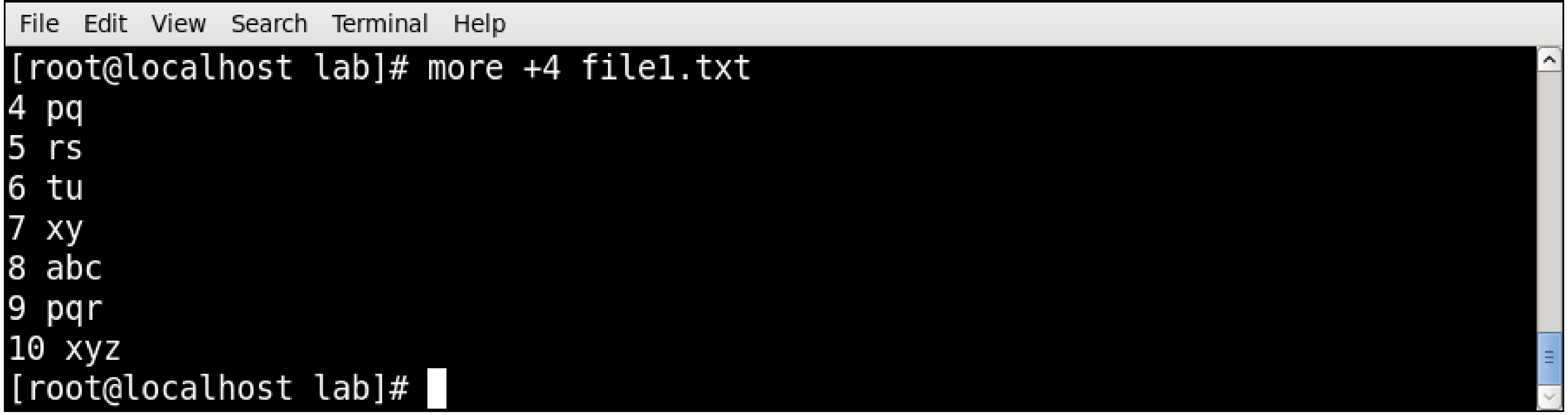
```
[root@localhost lab]# more file1.txt
1 ab
2 cd
3 ef
4 pq
5 rs
6 tu
7 xy
8 abc
9 pqr
10 xyz
[root@localhost lab]# 
```

more Command Example

more -number	**To Specify how many lines are printed in the screen for a given file**



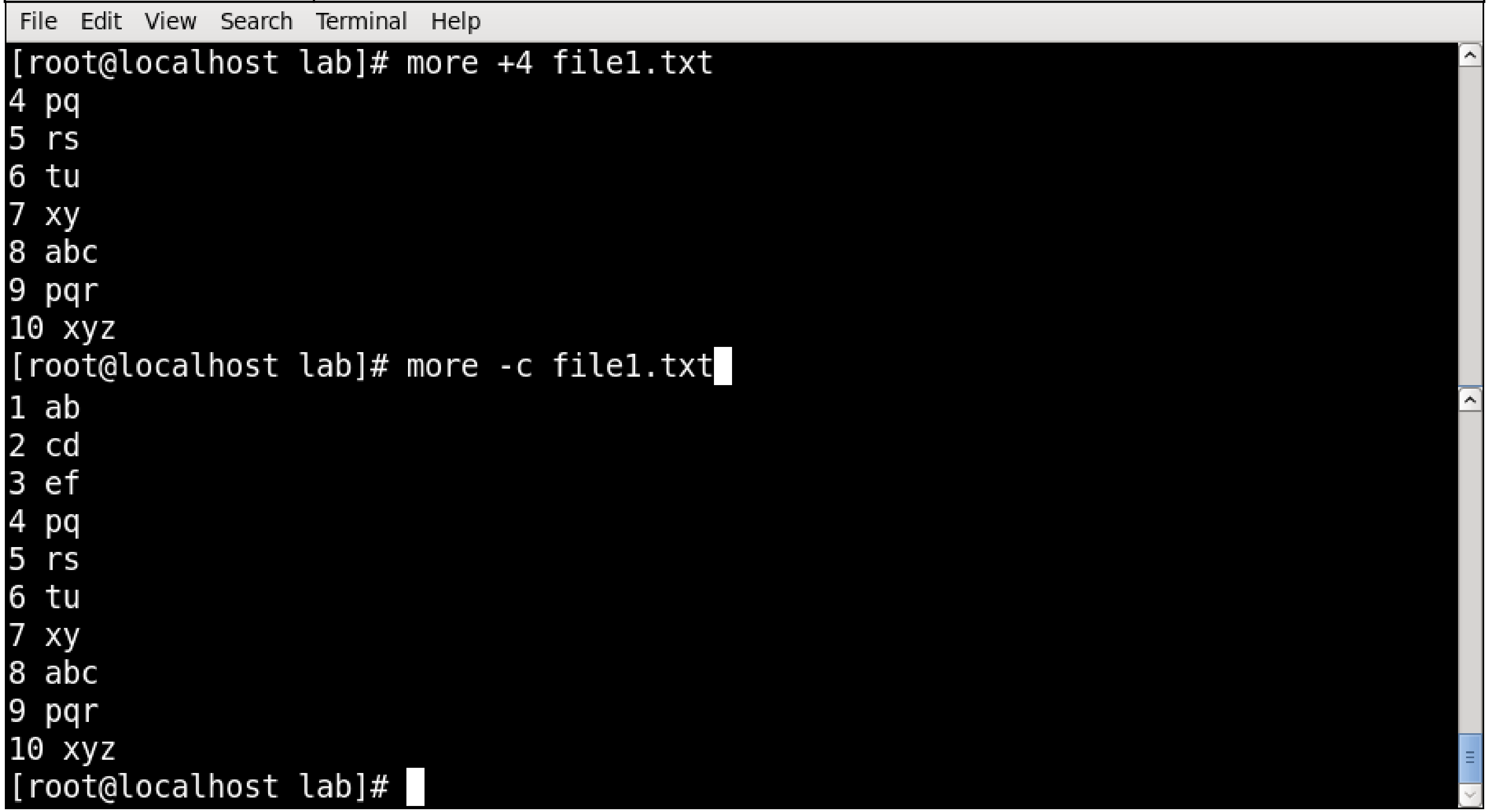
```
[root@localhost lab]# more -4 file1.txt
1 ab
2 cd
3 ef
4 pq
--More--(37%)
```



```
[root@localhost lab]# more +4 file1.txt
4 pq
5 rs
6 tu
7 xy
8 abc
9 pqr
10 xyz
[root@localhost lab]# 
```

more Command Example

more -c	**Clear screen before displaying**



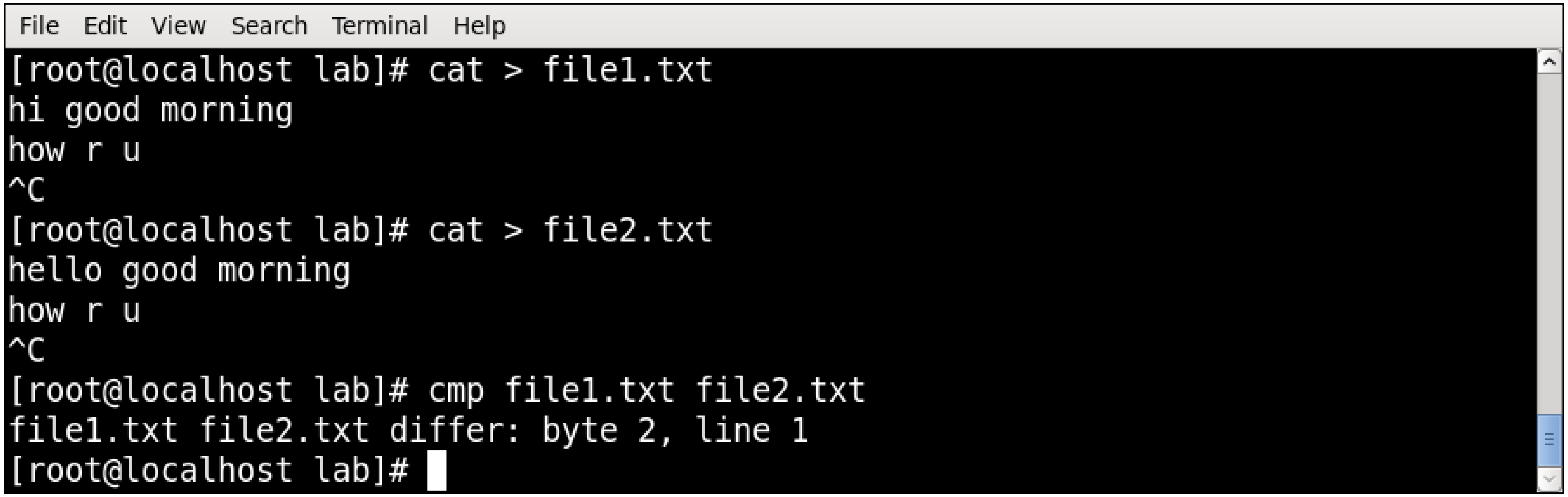
```
[root@localhost lab]# more +4 file1.txt
4 pq
5 rs
6 tu
7 xy
8 abc
9 pqr
10 xyz
[root@localhost lab]# more -c file1.txt
1 ab
2 cd
3 ef
4 pq
5 rs
6 tu
7 xy
8 abc
9 pqr
10 xyz
[root@localhost lab]#
```

cmp Command

- **cmp** command in Linux/UNIX is used to compare the two files byte by byte and helps you to find out whether the two files are identical or not.
- If a difference is found, it reports the byte and line number where the first difference is found.
- If no differences are found, by default, cmp returns no output.
- **Syntax :**
  - cmp [OPTION]... FILE1 [FILE2 [SKIP1 [SKIP2]]]

cmp Command Example

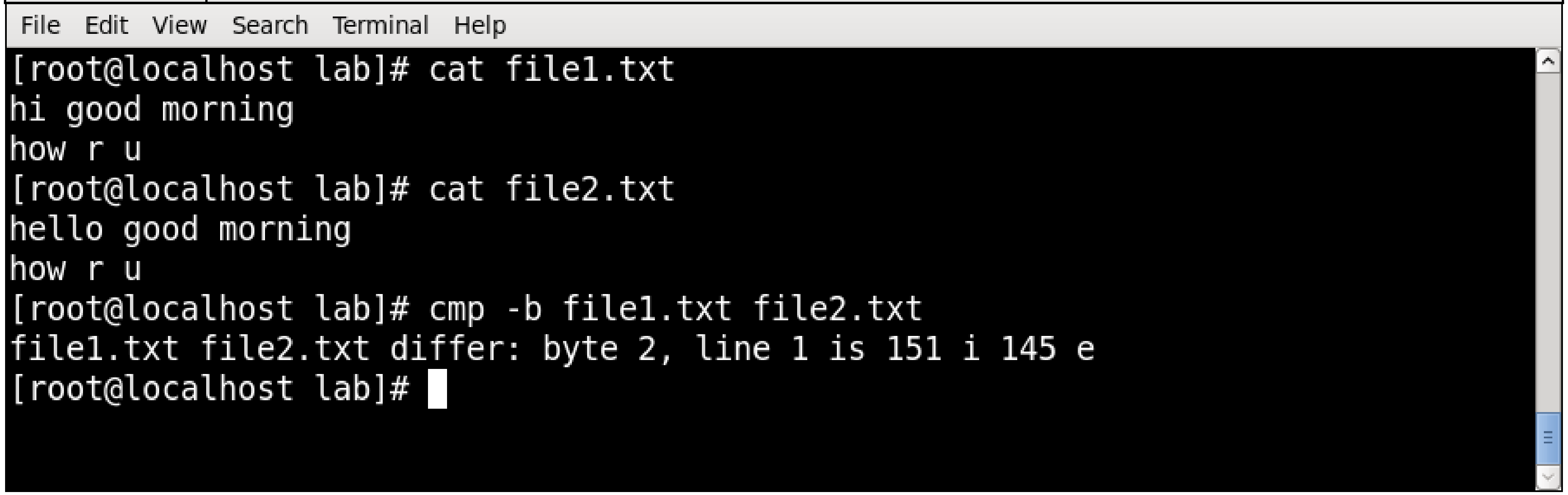
Option	Use
cmp -b	Print differing bytes
cmp -i	Skip a particular number of initial bytes from both the files
cmp -s	Do not print anything; only return an exit status indicating whether the files differ
cmp -n	Compare at most LIMIT bytes
cmp -l	Print byte position and byte value for all differing bytes



```
[root@localhost lab]# cat > file1.txt
hi good morning
how r u
^C
[root@localhost lab]# cat > file2.txt
hello good morning
how r u
^C
[root@localhost lab]# cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 2, line 1
[root@localhost lab]# 
```

cmp Command Example

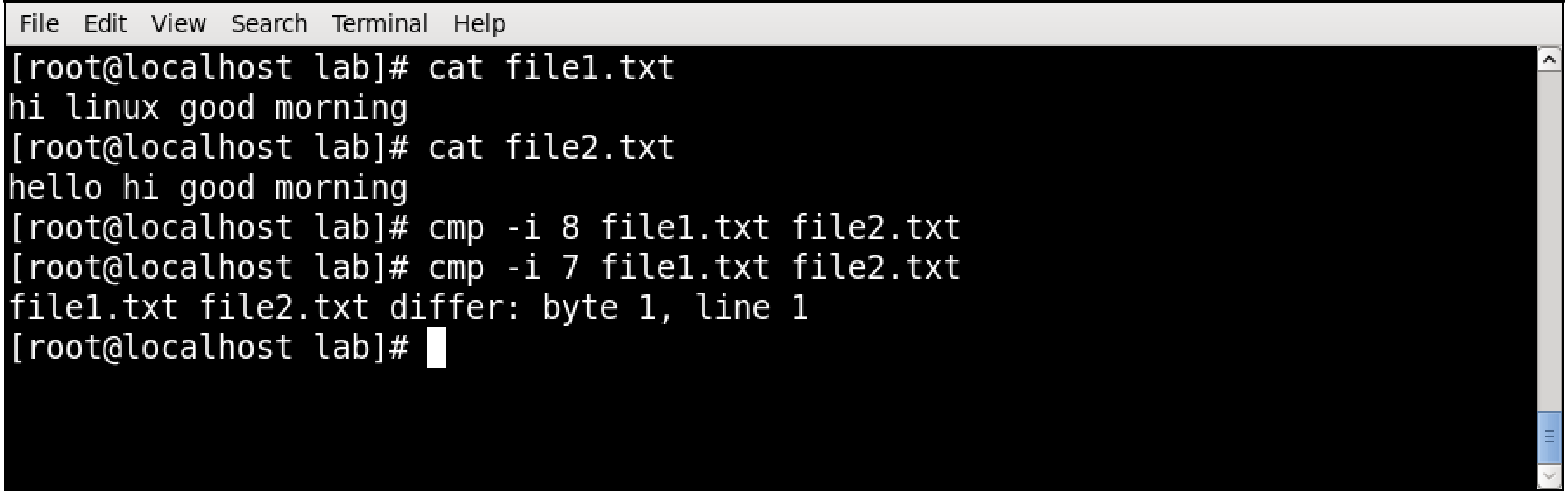
cmp -b	**Print differing bytes**



```
[root@localhost lab]# cat file1.txt
hi good morning
how r u
[root@localhost lab]# cat file2.txt
hello good morning
how r u
[root@localhost lab]# cmp -b file1.txt file2.txt
file1.txt file2.txt differ: byte 2, line 1 is 151 i 145 e
[root@localhost lab]#
```

cmp Command Example

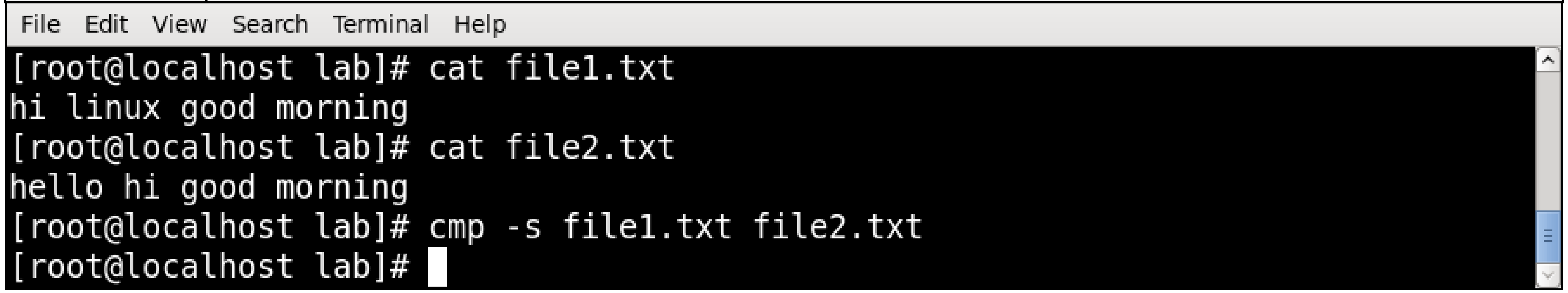
cmp -i	**Skip a particular number of initial bytes from both the files**



```
[root@localhost lab]# cat file1.txt
hi linux good morning
[root@localhost lab]# cat file2.txt
hello hi good morning
[root@localhost lab]# cmp -i 8 file1.txt file2.txt
[root@localhost lab]# cmp -i 7 file1.txt file2.txt
file1.txt file2.txt differ: byte 1, line 1
[root@localhost lab]# 
```

cmp Command Example

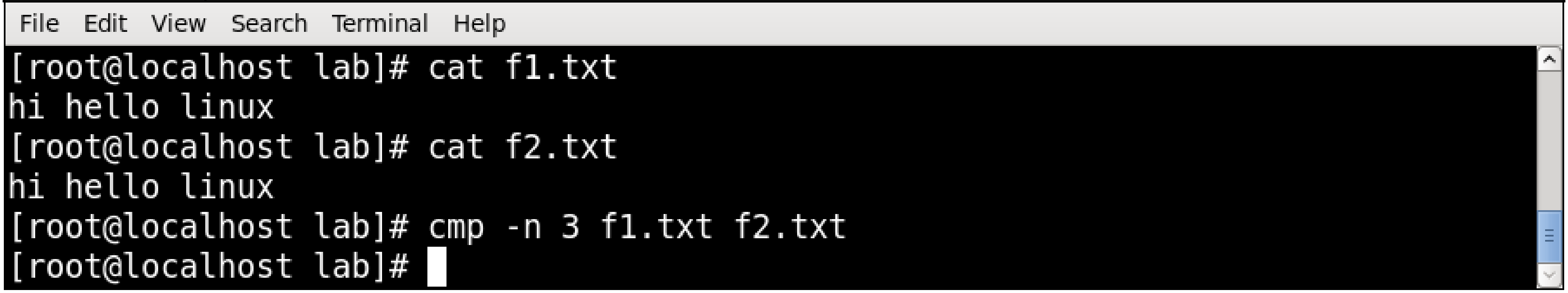
cmp -s	**Do not print anything; only return an exit status indicating whether the files differ**



```
[root@localhost lab]# cat file1.txt
hi linux good morning
[root@localhost lab]# cat file2.txt
hello hi good morning
[root@localhost lab]# cmp -s file1.txt file2.txt
[root@localhost lab]# 
```

cmp Command Example

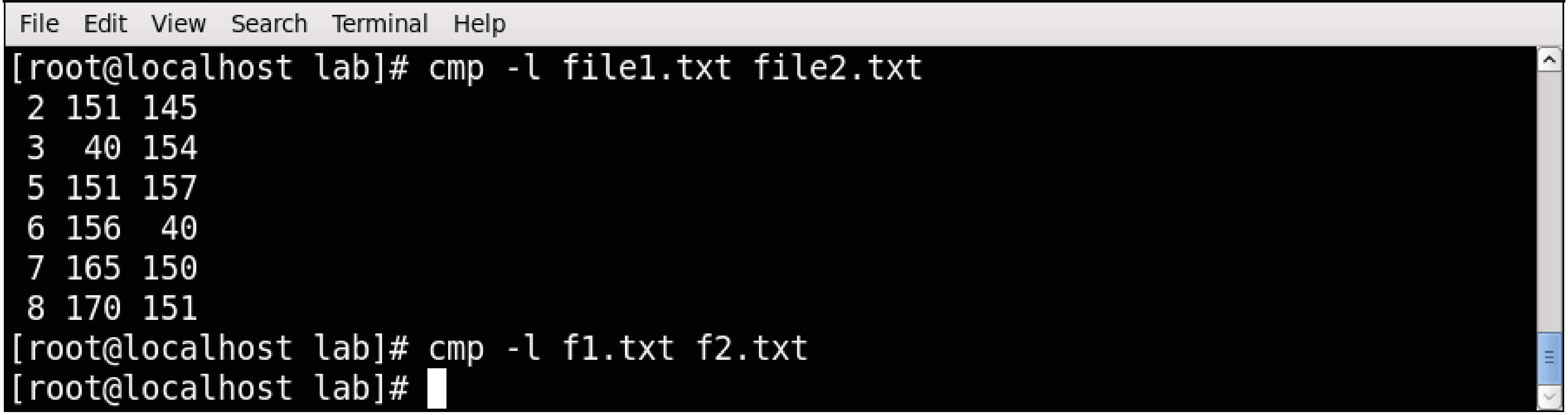
cmp -n	**Compare at most LIMIT bytes**



```
[root@localhost lab]# cat f1.txt
hi hello linux
[root@localhost lab]# cat f2.txt
hi hello linux
[root@localhost lab]# cmp -n 3 f1.txt f2.txt
[root@localhost lab]# 
```

cmp Command Example

cmp -l	**Print byte position and byte value for all differing bytes**



```
[root@localhost lab]# cmp -l file1.txt file2.txt
 2 151 145
 3  40 154
 5 151 157
 6 156  40
 7 165 150
 8 170 151
[root@localhost lab]# cmp -l f1.txt f2.txt
[root@localhost lab]# 
```

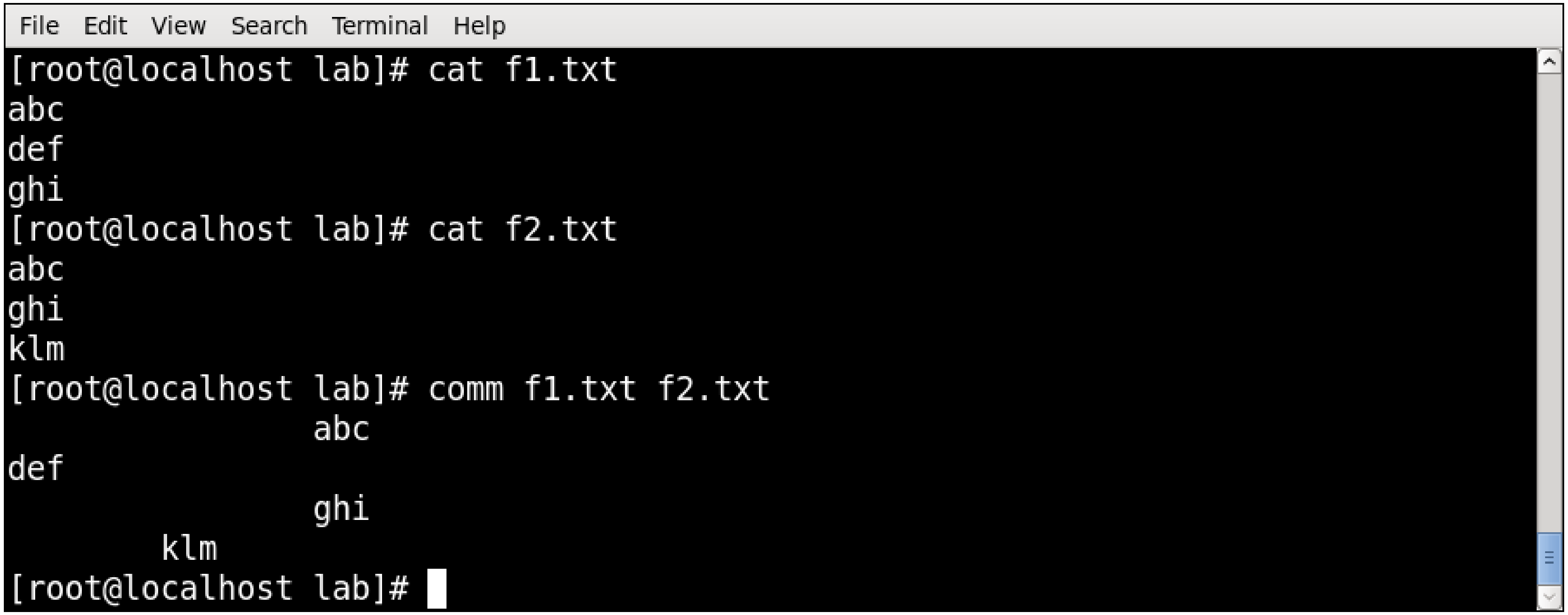
comm Command

- Compare two sorted files line by line.

- **Syntax :**

  comm [OPTION]... FILE1 FILE2

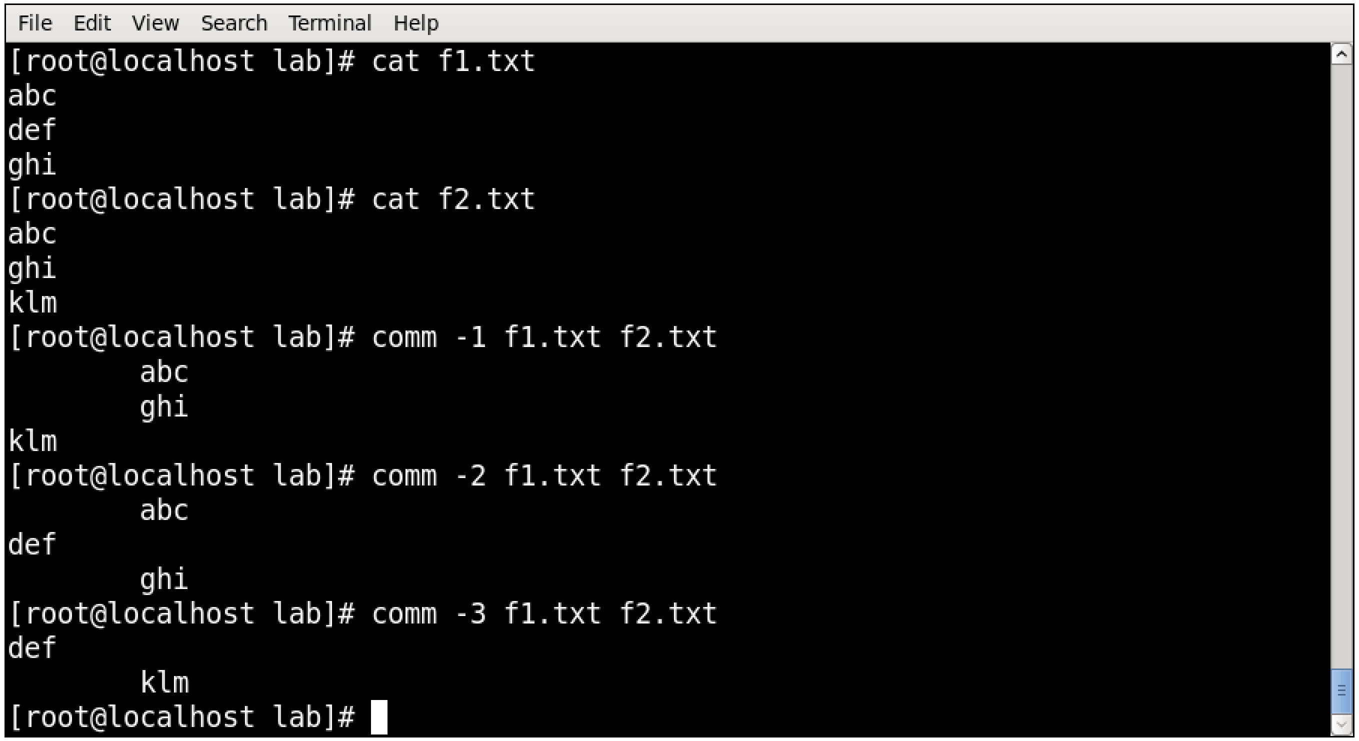
- **Example :**



```
[root@localhost lab]# cat f1.txt
abc
def
ghi
[root@localhost lab]# cat f2.txt
abc
ghi
klm
[root@localhost lab]# comm f1.txt f2.txt
                abc
def
                ghi
        klm
[root@localhost lab]# 
```

comm Command Example

Option	Use
comm -1	Suppress column 1 (lines unique to FILE1)
comm -2	Suppress column 2 (lines unique to FILE2)
comm -3	Suppress column 3 (lines that appear in both files)



```
[root@localhost lab]# cat f1.txt
abc
def
ghi
[root@localhost lab]# cat f2.txt
abc
ghi
klm
[root@localhost lab]# comm -1 f1.txt f2.txt
        abc
        ghi
klm
[root@localhost lab]# comm -2 f1.txt f2.txt
        abc
def
        ghi
[root@localhost lab]# comm -3 f1.txt f2.txt
def
        klm
[root@localhost lab]# 
```

diff(difference) Command

- This command is used to display the differences in the files by comparing the files line by line
- diff analyzes two files and prints the lines that are different. Essentially, it outputs a set of instructions for how to change one file to make it identical to the second file.

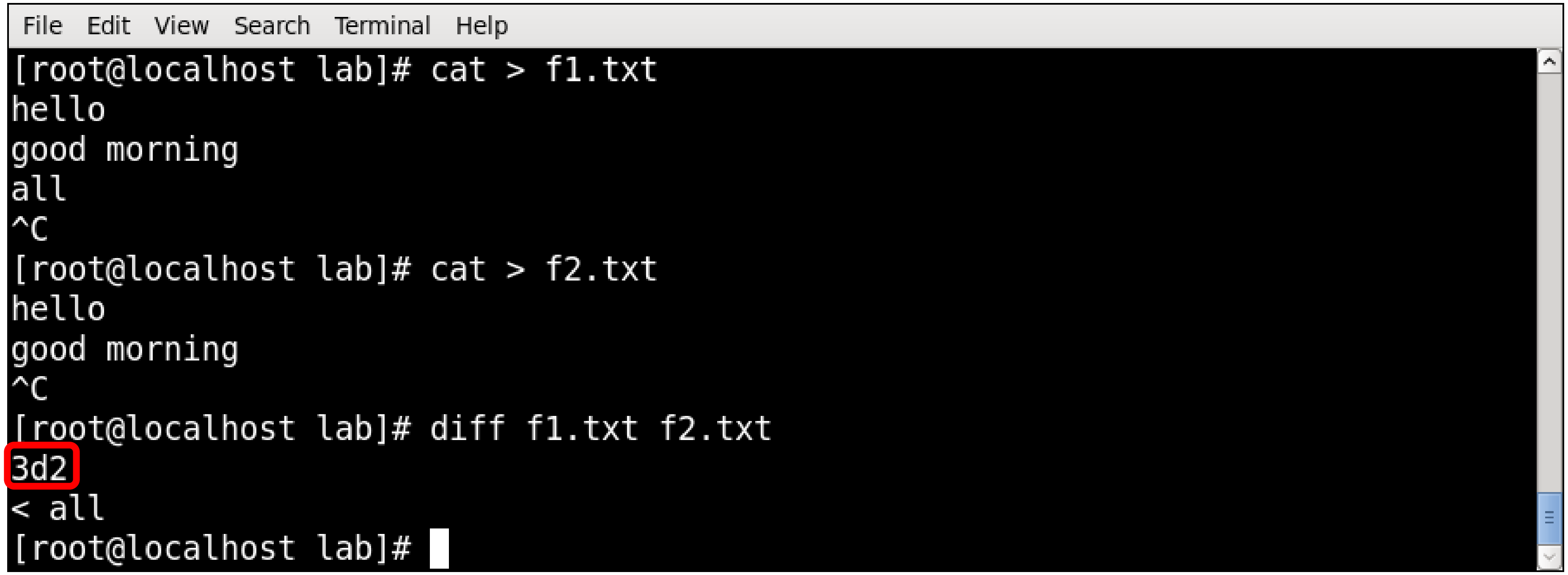
- **Syntax :**

```
diff [options] File1 File2
```

- **Example :**

Option	Use
diff -b	Ignores spacing differences
diff -i	Ignores case

diff Command Example



```
[root@localhost lab]# cat > f1.txt
hello
good morning
all
^C
[root@localhost lab]# cat > f2.txt
hello
good morning
^C
[root@localhost lab]# diff f1.txt f2.txt
3d2
< all
[root@localhost lab]# 
```

Special symbols are:

- a : add
- c : change
- d : delete

chmod(change mode) Command

- chmod is used to change the permissions of files or directories.
- **Syntax :**

  chmod [reference][operator][mode] file...
- **Example :**

Reference	Class	Description
u	owner	file's owner
g	group	users who are members of the file's group
o	others	users who are neither the file's owner nor members of the file's group
a	all	All three of the above

chmod(change mode) Command

Operator	Description
+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the specified classes

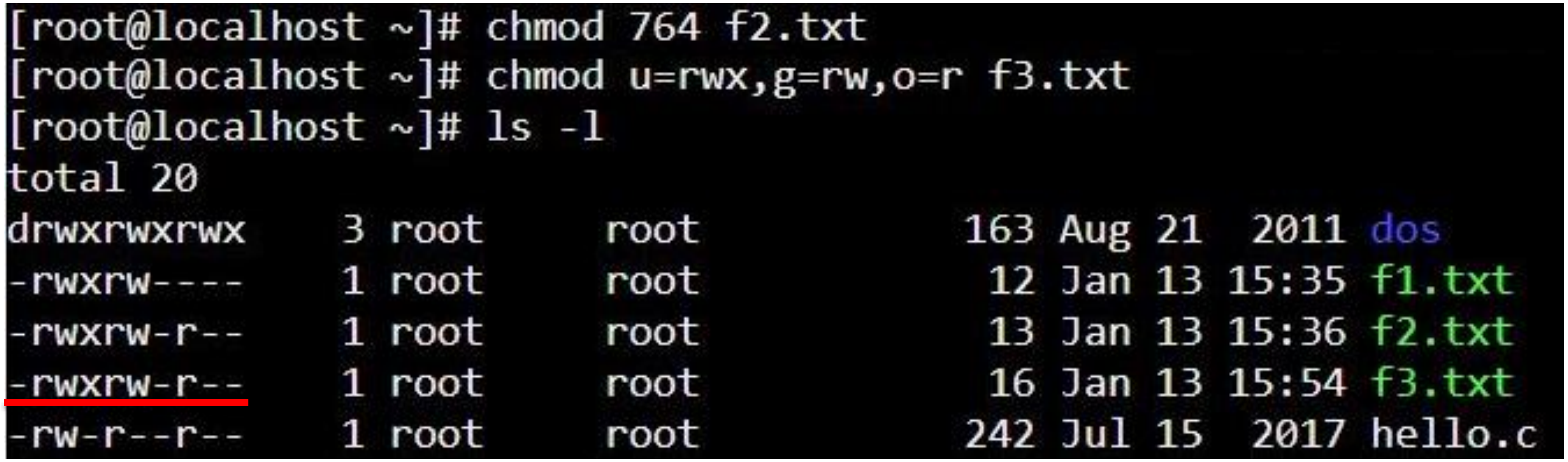
Permission	Description
r	Permission to read the file
w	Permission to write (or delete) the file
x	Permission to execute the file, or, in the case of a directory, search it

chmod Command Example

- Each write, read, and execute permissions have following number value:

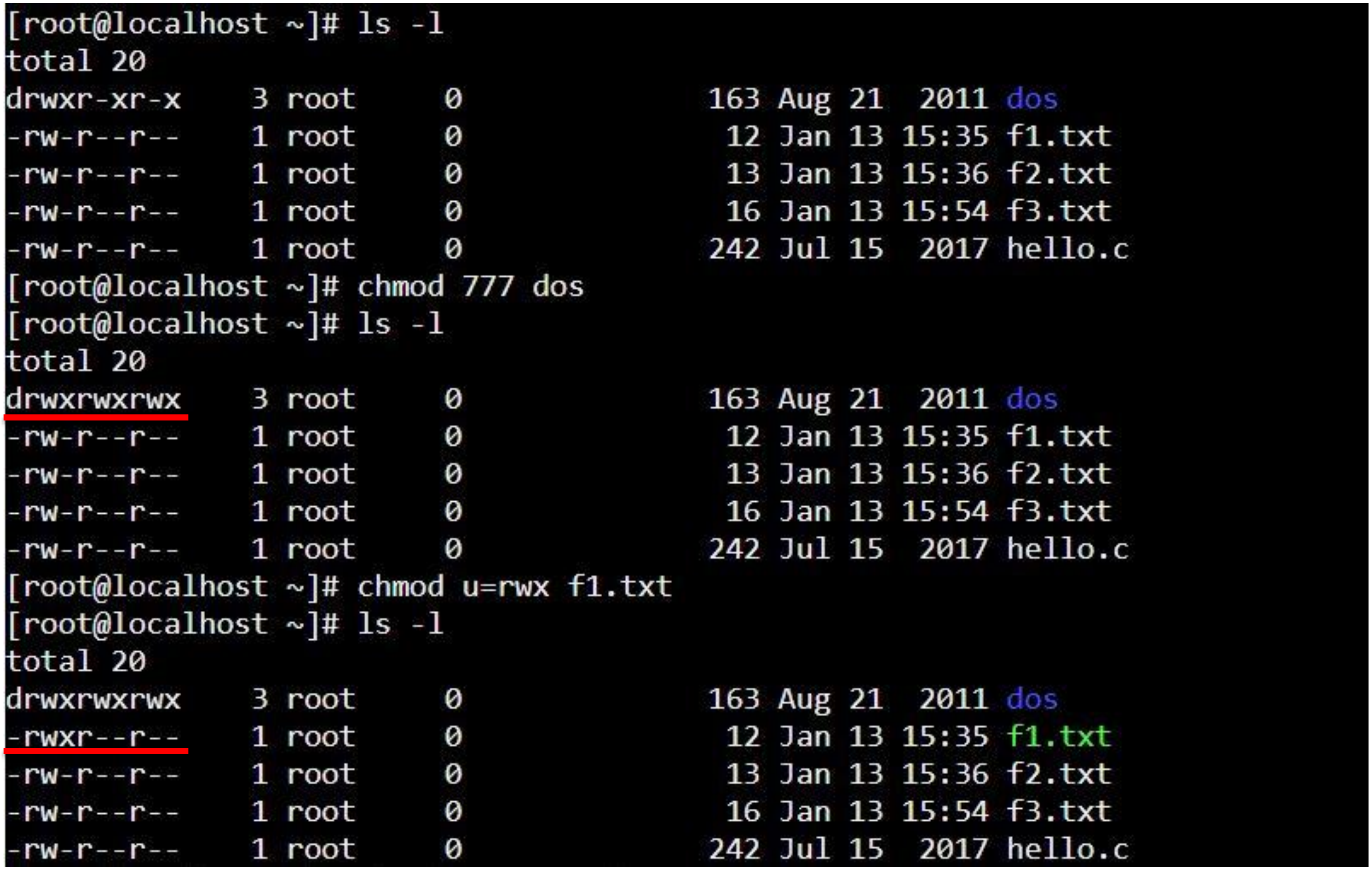
u	owner
g	group
o	others
a	all

r (read)	4
w (write)	2
x (execute)	1
no permissions	0



```
[root@localhost ~]# chmod 764 f2.txt
[root@localhost ~]# chmod u=rwx,g=rw,o=r f3.txt
[root@localhost ~]# ls -l
total 20
drwxrwxrwx    3 root      root             163 Aug 21  2011 dos
-rwxrw----    1 root      root              12 Jan 13 15:35 f1.txt
-rwxrw-r--    1 root      root              13 Jan 13 15:36 f2.txt
-rwxrw-r--    1 root      root              16 Jan 13 15:54 f3.txt
-rw-r--r--    1 root      root             242 Jul 15  2017 hello.c
```

chmod Command Example



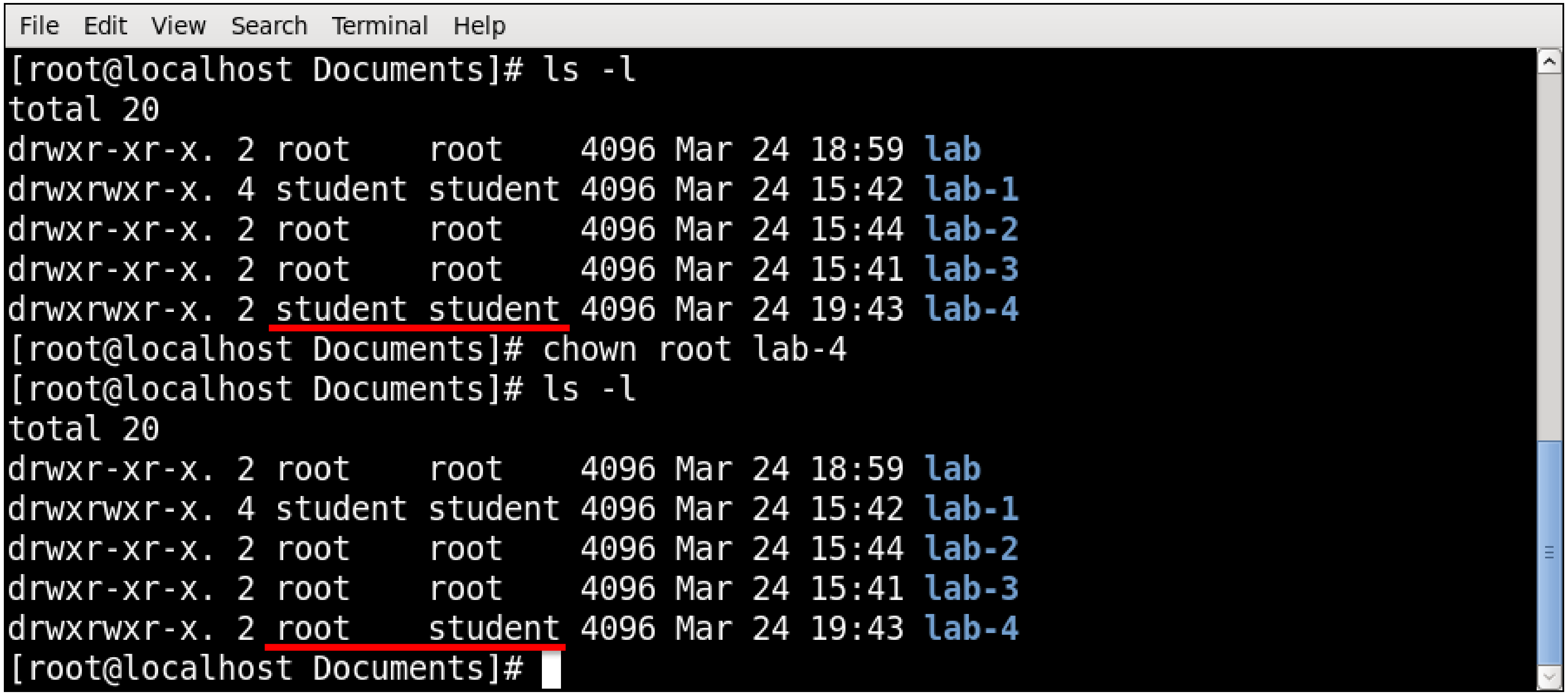
```
[root@localhost ~]# ls -l
total 20
drwxr-xr-x    3 root     0              163 Aug 21  2011 dos
-rw-r--r--    1 root     0               12 Jan 13 15:35 f1.txt
-rw-r--r--    1 root     0               13 Jan 13 15:36 f2.txt
-rw-r--r--    1 root     0               16 Jan 13 15:54 f3.txt
-rw-r--r--    1 root     0              242 Jul 15  2017 hello.c
[root@localhost ~]# chmod 777 dos
[root@localhost ~]# ls -l
total 20
drwxrwxrwx    3 root     0              163 Aug 21  2011 dos
-rw-r--r--    1 root     0               12 Jan 13 15:35 f1.txt
-rw-r--r--    1 root     0               13 Jan 13 15:36 f2.txt
-rw-r--r--    1 root     0               16 Jan 13 15:54 f3.txt
-rw-r--r--    1 root     0              242 Jul 15  2017 hello.c
[root@localhost ~]# chmod u=rwx f1.txt
[root@localhost ~]# ls -l
total 20
drwxrwxrwx    3 root     0              163 Aug 21  2011 dos
-rwxr--r--    1 root     0               12 Jan 13 15:35 f1.txt
-rw-r--r--    1 root     0               13 Jan 13 15:36 f2.txt
-rw-r--r--    1 root     0               16 Jan 13 15:54 f3.txt
-rw-r--r--    1 root     0              242 Jul 15  2017 hello.c
```

chown(change owner) Command

- The chown command changes ownership of files and directories in a Linux filesystem.
- **Syntax :**

  ```
  chown [OPTIONS] USER[:GROUP] FILE(s)
  ```

chown Command Example



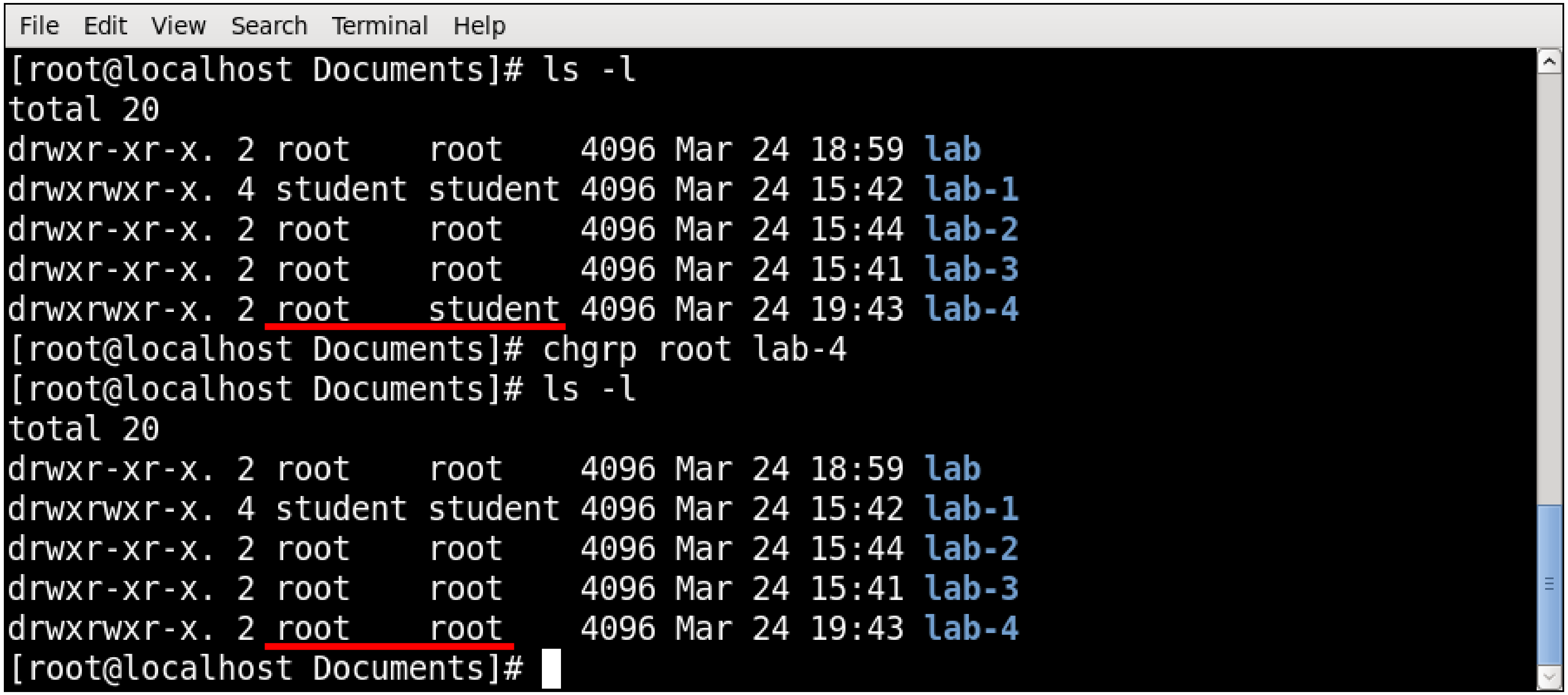
```
[root@localhost Documents]# ls -l
total 20
drwxr-xr-x. 2 root    root    4096 Mar 24 18:59 lab
drwxrwxr-x. 4 student student 4096 Mar 24 15:42 lab-1
drwxr-xr-x. 2 root    root    4096 Mar 24 15:44 lab-2
drwxr-xr-x. 2 root    root    4096 Mar 24 15:41 lab-3
drwxrwxr-x. 2 student student 4096 Mar 24 19:43 lab-4
[root@localhost Documents]# chown root lab-4
[root@localhost Documents]# ls -l
total 20
drwxr-xr-x. 2 root    root    4096 Mar 24 18:59 lab
drwxrwxr-x. 4 student student 4096 Mar 24 15:42 lab-1
drwxr-xr-x. 2 root    root    4096 Mar 24 15:44 lab-2
drwxr-xr-x. 2 root    root    4096 Mar 24 15:41 lab-3
drwxrwxr-x. 2 root    student 4096 Mar 24 19:43 lab-4
[root@localhost Documents]# 
```

chgrp(change group) Command

- The chgrp command is used to change group ownership of a file/directory.
- **Syntax :**

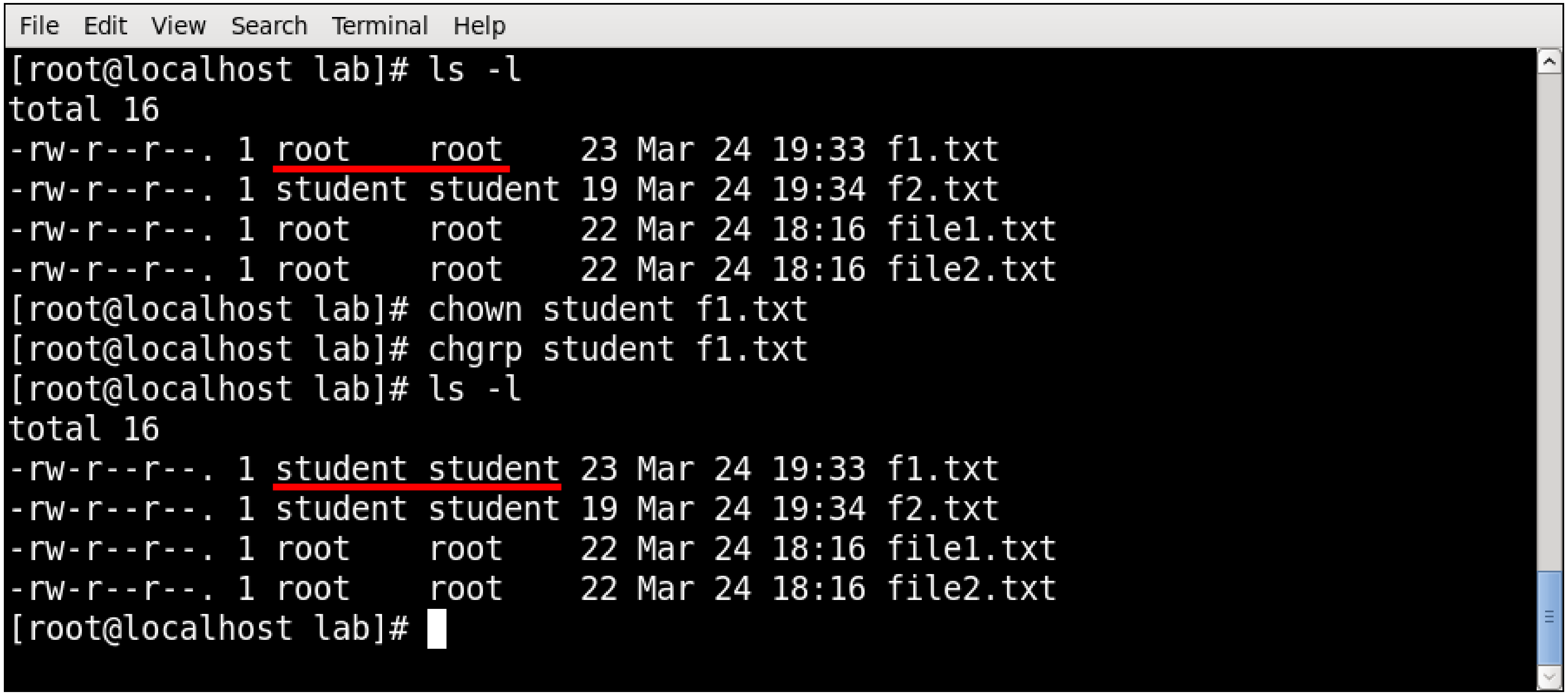
  chgrp [OPTION]… GROUP FILE/DIR…

chgrp Command Example



```
[root@localhost Documents]# ls -l
total 20
drwxr-xr-x. 2 root    root    4096 Mar 24 18:59 lab
drwxrwxr-x. 4 student student 4096 Mar 24 15:42 lab-1
drwxr-xr-x. 2 root    root    4096 Mar 24 15:44 lab-2
drwxr-xr-x. 2 root    root    4096 Mar 24 15:41 lab-3
drwxrwxr-x. 2 root    student 4096 Mar 24 19:43 lab-4
[root@localhost Documents]# chgrp root lab-4
[root@localhost Documents]# ls -l
total 20
drwxr-xr-x. 2 root    root    4096 Mar 24 18:59 lab
drwxrwxr-x. 4 student student 4096 Mar 24 15:42 lab-1
drwxr-xr-x. 2 root    root    4096 Mar 24 15:44 lab-2
drwxr-xr-x. 2 root    root    4096 Mar 24 15:41 lab-3
drwxrwxr-x. 2 root    root    4096 Mar 24 19:43 lab-4
[root@localhost Documents]#
```

chgrp Command Example

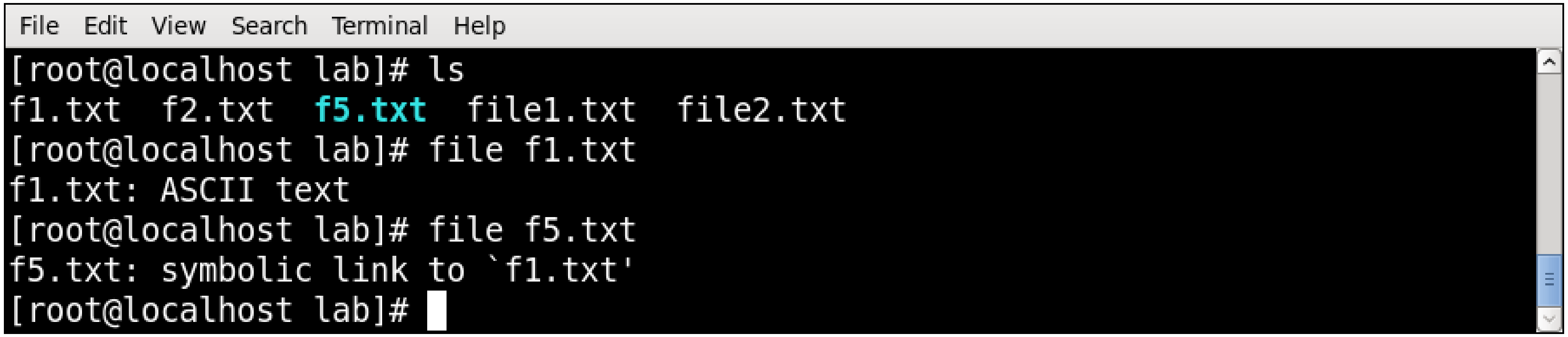


```
[root@localhost lab]# ls -l
total 16
-rw-r--r--. 1 root    root    23 Mar 24 19:33 f1.txt
-rw-r--r--. 1 student student 19 Mar 24 19:34 f2.txt
-rw-r--r--. 1 root    root    22 Mar 24 18:16 file1.txt
-rw-r--r--. 1 root    root    22 Mar 24 18:16 file2.txt
[root@localhost lab]# chown student f1.txt
[root@localhost lab]# chgrp student f1.txt
[root@localhost lab]# ls -l
total 16
-rw-r--r--. 1 student student 23 Mar 24 19:33 f1.txt
-rw-r--r--. 1 student student 19 Mar 24 19:34 f2.txt
-rw-r--r--. 1 root    root    22 Mar 24 18:16 file1.txt
-rw-r--r--. 1 root    root    22 Mar 24 18:16 file2.txt
[root@localhost lab]# 
```

file Command

- The file command is used to determine a file's type.
- **Syntax :**

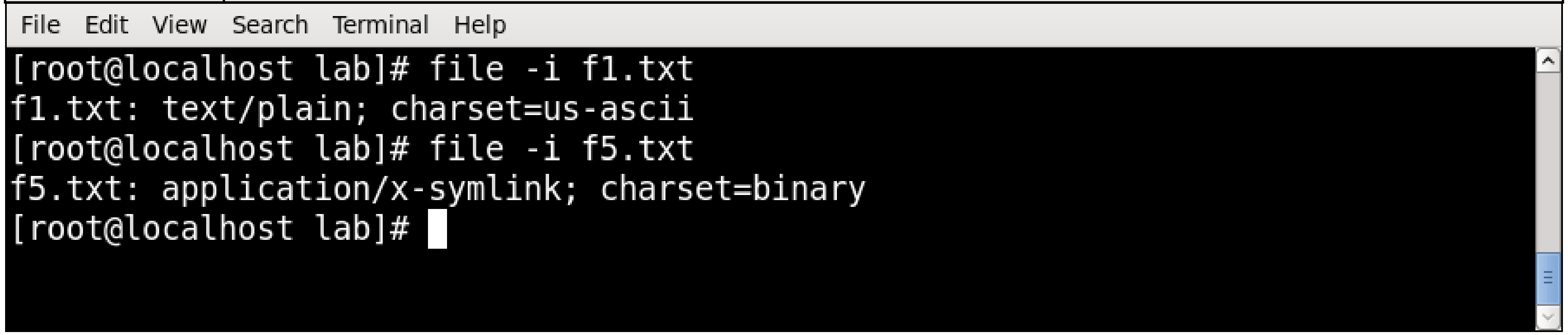
  file [OPTIONS] file1 file2 ...
- **Example :**



```
[root@localhost lab]# ls
f1.txt  f2.txt  f5.txt  file1.txt  file2.txt
[root@localhost lab]# file f1.txt
f1.txt: ASCII text
[root@localhost lab]# file f5.txt
f5.txt: symbolic link to `f1.txt'
[root@localhost lab]# 
```

file Command Example

file -i	**To view the mime type of a file rather than the human readable format**

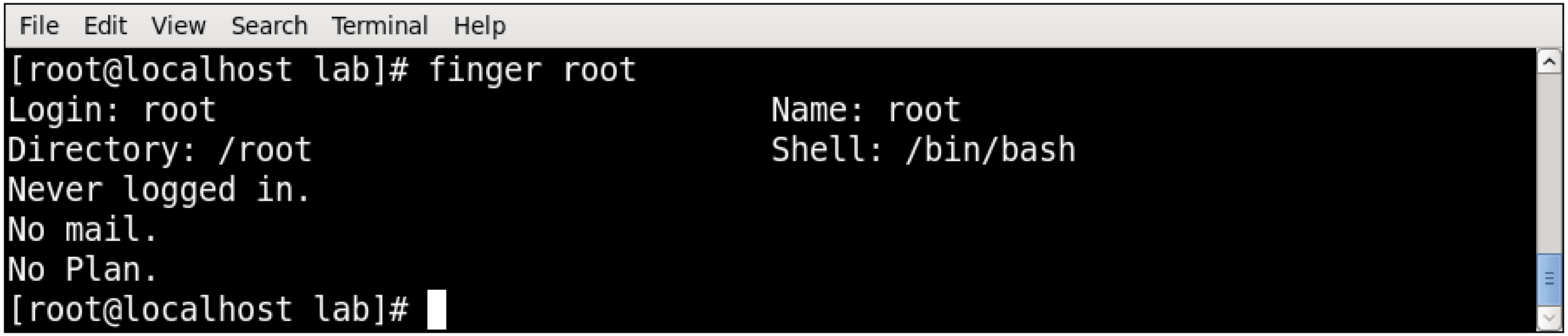


```
[root@localhost lab]# file -i f1.txt
f1.txt: text/plain; charset=us-ascii
[root@localhost lab]# file -i f5.txt
f5.txt: application/x-symlink; charset=binary
[root@localhost lab]#
```

finger Command

- finger looks up and displays information about system users.
- **Syntax :**

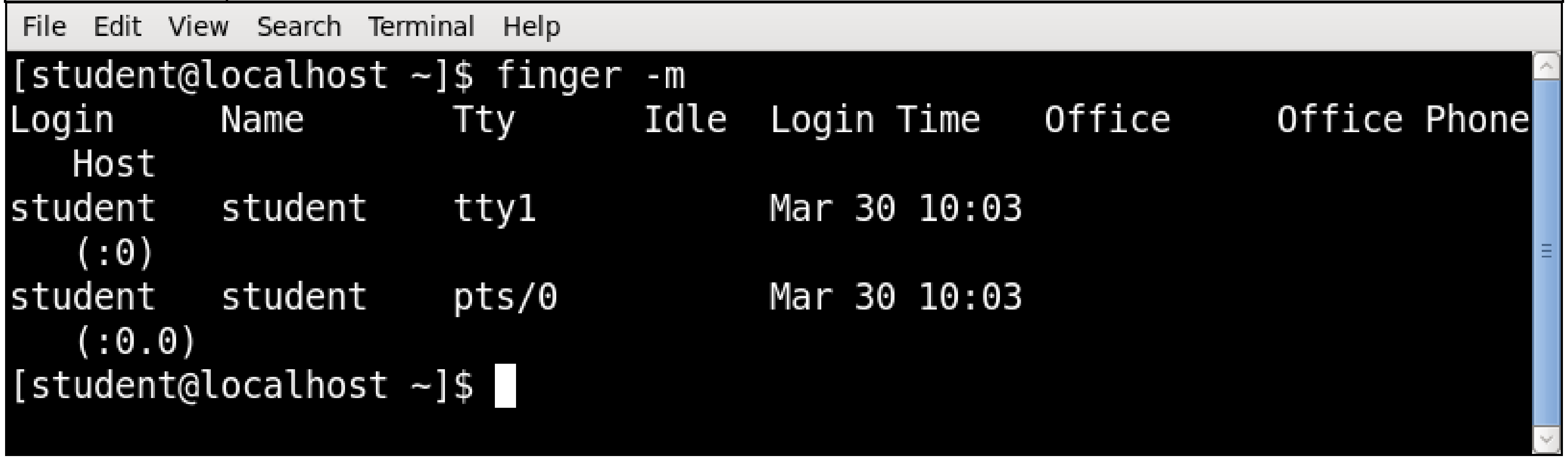
  finger [-option] [username]
- **Example :**



```
[root@localhost lab]# finger root
Login: root                             Name: root
Directory: /root                        Shell: /bin/bash
Never logged in.
No mail.
No Plan.
[root@localhost lab]# 
```

finger Command Example

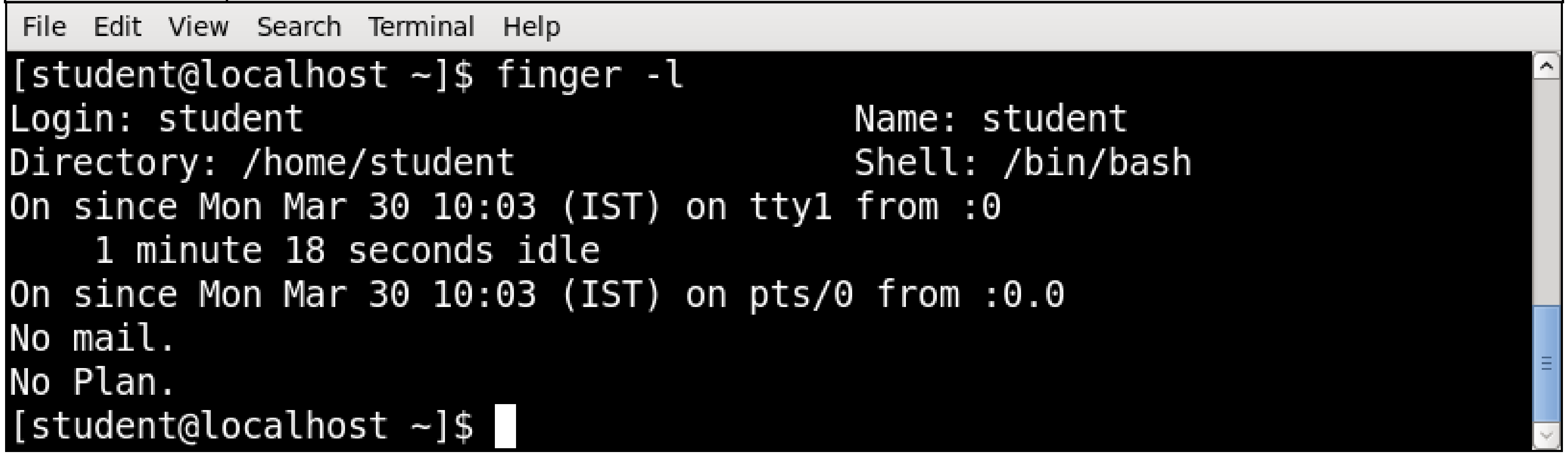
finger -m	**Match arguments only on user name (not first or last name)**



```
[student@localhost ~]$ finger -m
Login     Name       Tty       Idle  Login Time   Office     Office Phone
   Host
student   student    tty1            Mar 30 10:03
   (:0)
student   student    pts/0           Mar 30 10:03
   (:0.0)
[student@localhost ~]$ 
```

finger Command Example

finger -l	**Force long output format**



```
[student@localhost ~]$ finger -l
Login: student                          Name: student
Directory: /home/student                Shell: /bin/bash
On since Mon Mar 30 10:03 (IST) on tty1 from :0
    1 minute 18 seconds idle
On since Mon Mar 30 10:03 (IST) on pts/0 from :0.0
No mail.
No Plan.
[student@localhost ~]$ 
```

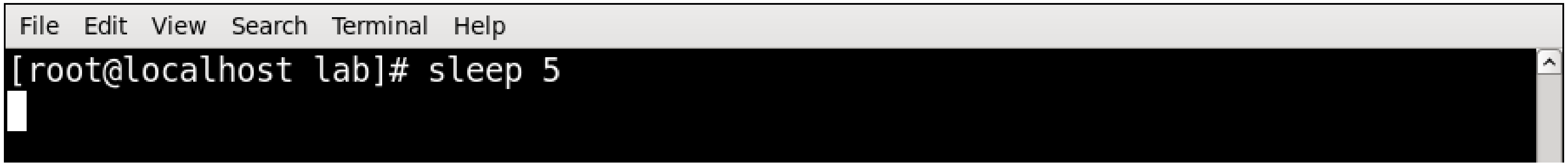
sleep Command

- The sleep command is used to delay for a specified amount of time.

- **Syntax :**

  sleep NUMBER[SUFFIX]...

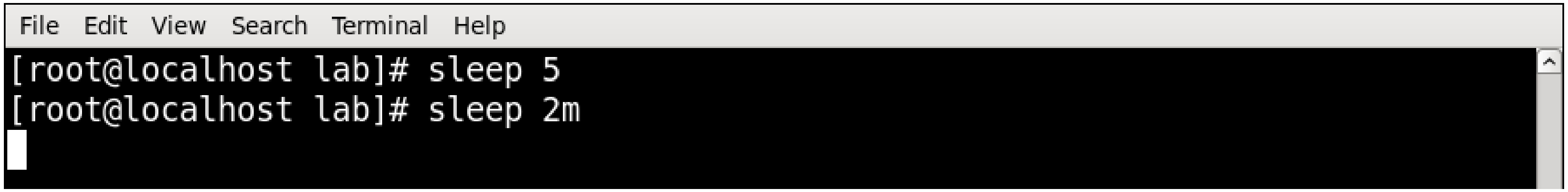
- **Example :**



```
[root@localhost lab]# sleep 5
```

sleep Command Example

- s for seconds; this is a default one if you don't specify any letter after the integer.
- m for minutes.
- h for hours.
- d for days.



```
[root@localhost lab]# sleep 5
[root@localhost lab]# sleep 2m
```

ps Command

- Reports a snapshot of the status of currently running processes.

- **Syntax :**

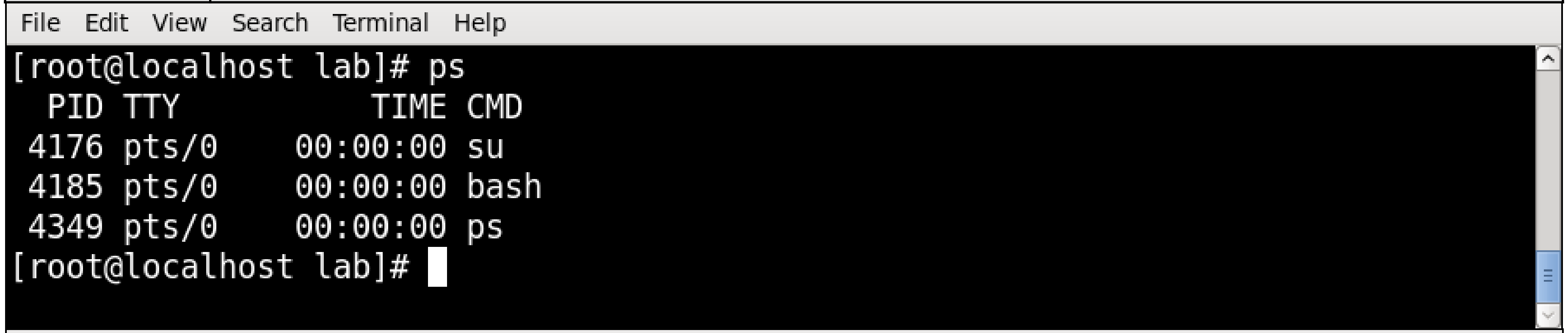
  ps [option]

- **Example :**

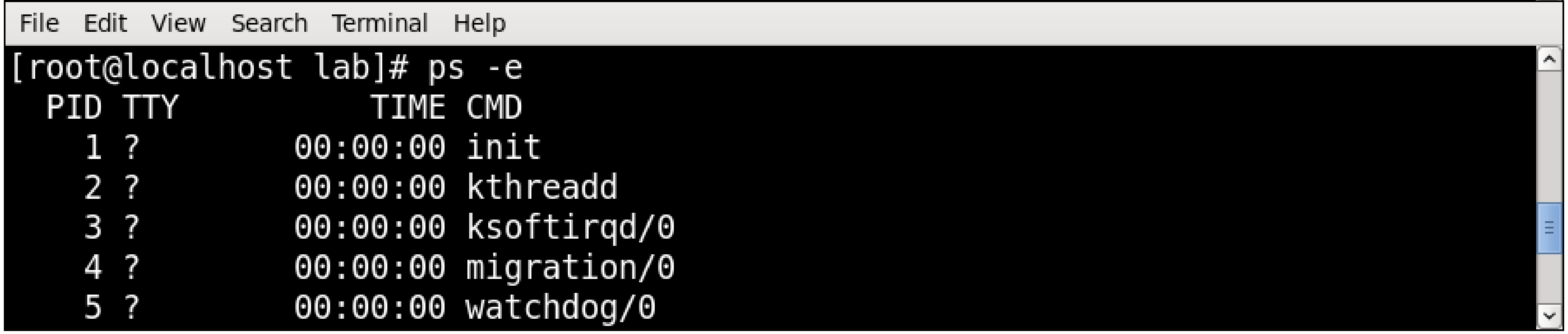
Option	Use
ps -e	Display every active process on a Linux system in generic (Unix/Linux) format
ps -x	View all processes owned by you
ps -f	To provide more information on processes
ps -u	Filter processes by its user

ps Command Example

ps -e	**Display every active process on a Linux system in generic (Unix/Linux) format**



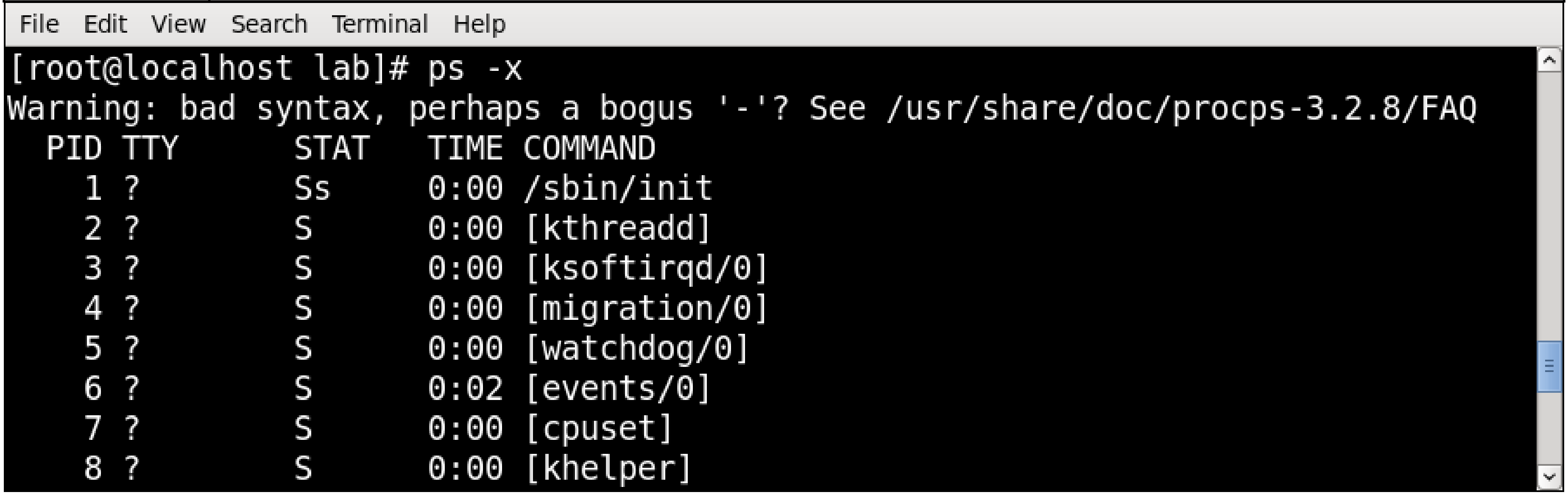
```
[root@localhost lab]# ps
  PID TTY          TIME CMD
 4176 pts/0    00:00:00 su
 4185 pts/0    00:00:00 bash
 4349 pts/0    00:00:00 ps
[root@localhost lab]# 
```



```
[root@localhost lab]# ps -e
  PID TTY          TIME CMD
    1 ?        00:00:00 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    4 ?        00:00:00 migration/0
    5 ?        00:00:00 watchdog/0
```

ps Command Example

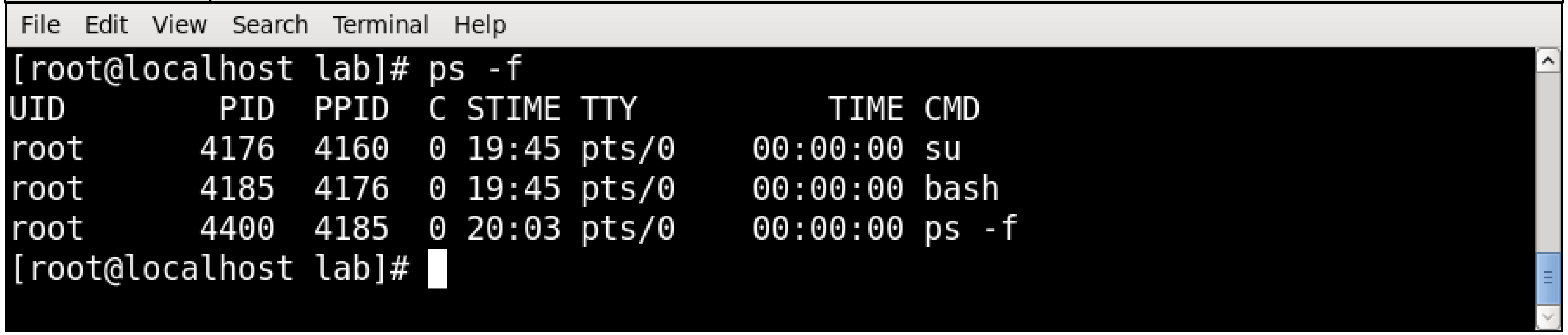
ps -x	**View all processes owned by you**



```
[root@localhost lab]# ps -x
Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 /sbin/init
    2 ?        S      0:00 [kthreadd]
    3 ?        S      0:00 [ksoftirqd/0]
    4 ?        S      0:00 [migration/0]
    5 ?        S      0:00 [watchdog/0]
    6 ?        S      0:02 [events/0]
    7 ?        S      0:00 [cpuset]
    8 ?        S      0:00 [khelper]
```

ps Command Example

ps -f	**To provide more information on processes**



```
[root@localhost lab]# ps -f
UID         PID  PPID  C STIME TTY          TIME CMD
root       4176  4160  0 19:45 pts/0    00:00:00 su
root       4185  4176  0 19:45 pts/0    00:00:00 bash
root       4400  4185  0 20:03 pts/0    00:00:00 ps -f
[root@localhost lab]# 
```

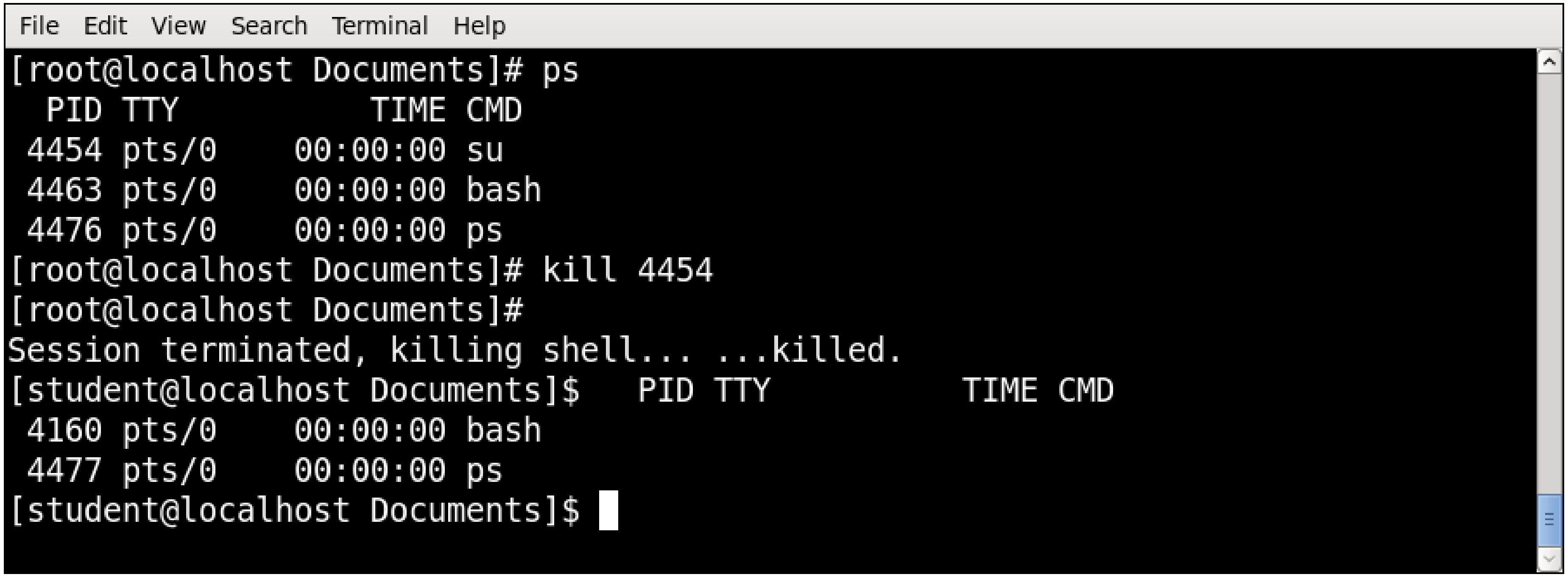
kill Command

- It is used to terminate processes manually.
- kill command sends a signal to a process which terminates the process.
- If the user doesn’t specify any signal which is to be sent along with kill command then default TERM signal is sent that terminates the process..

▪ **Syntax :**

```
kill [option] PID
```

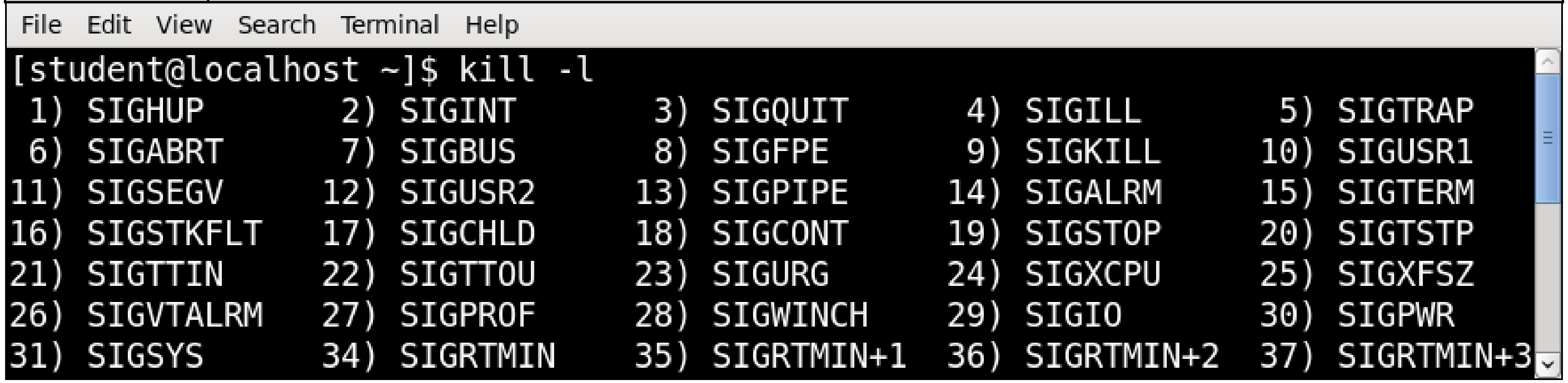
kill Command Example



```
[root@localhost Documents]# ps
  PID TTY          TIME CMD
 4454 pts/0    00:00:00 su
 4463 pts/0    00:00:00 bash
 4476 pts/0    00:00:00 ps
[root@localhost Documents]# kill 4454
[root@localhost Documents]#
Session terminated, killing shell... ...killed.
[student@localhost Documents]$   PID TTY          TIME CMD
 4160 pts/0    00:00:00 bash
 4477 pts/0    00:00:00 ps
[student@localhost Documents]$ 
```

kill Command Example

Kill -l	**To display all the available signals**



```
[student@localhost ~]$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
```

wc Command

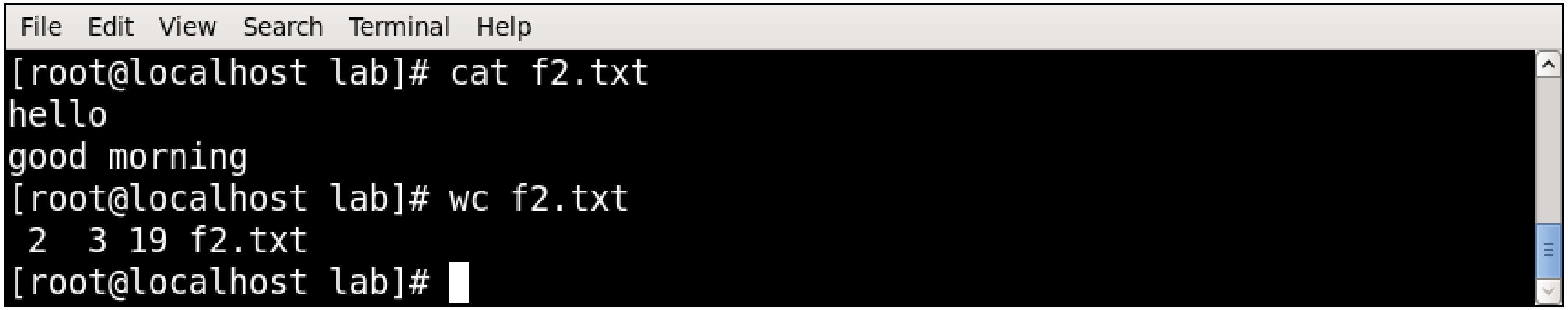
- It s used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments.

- **Syntax :**

  wc [options] filenames

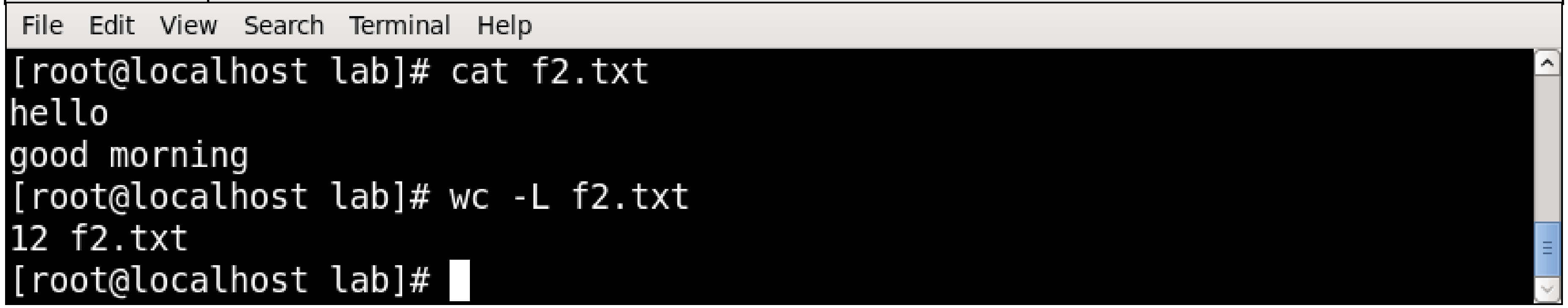
Option	**Use**
wc -l	Prints the number of lines in a file
wc -w	Prints the number of words in a file
wc -c	Displays the count of bytes in a file
wc -L	Prints only the length of the longest line in a file

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]# 
```

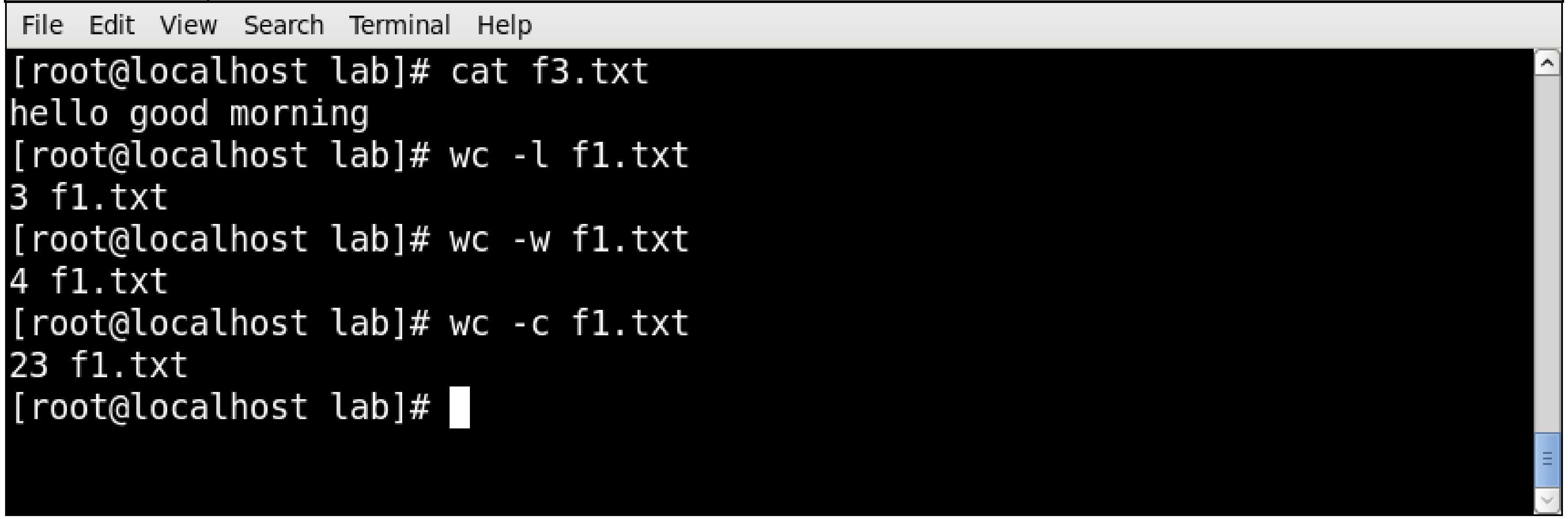
wc -L	**Prints only the length of the longest line in a file**



```
[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

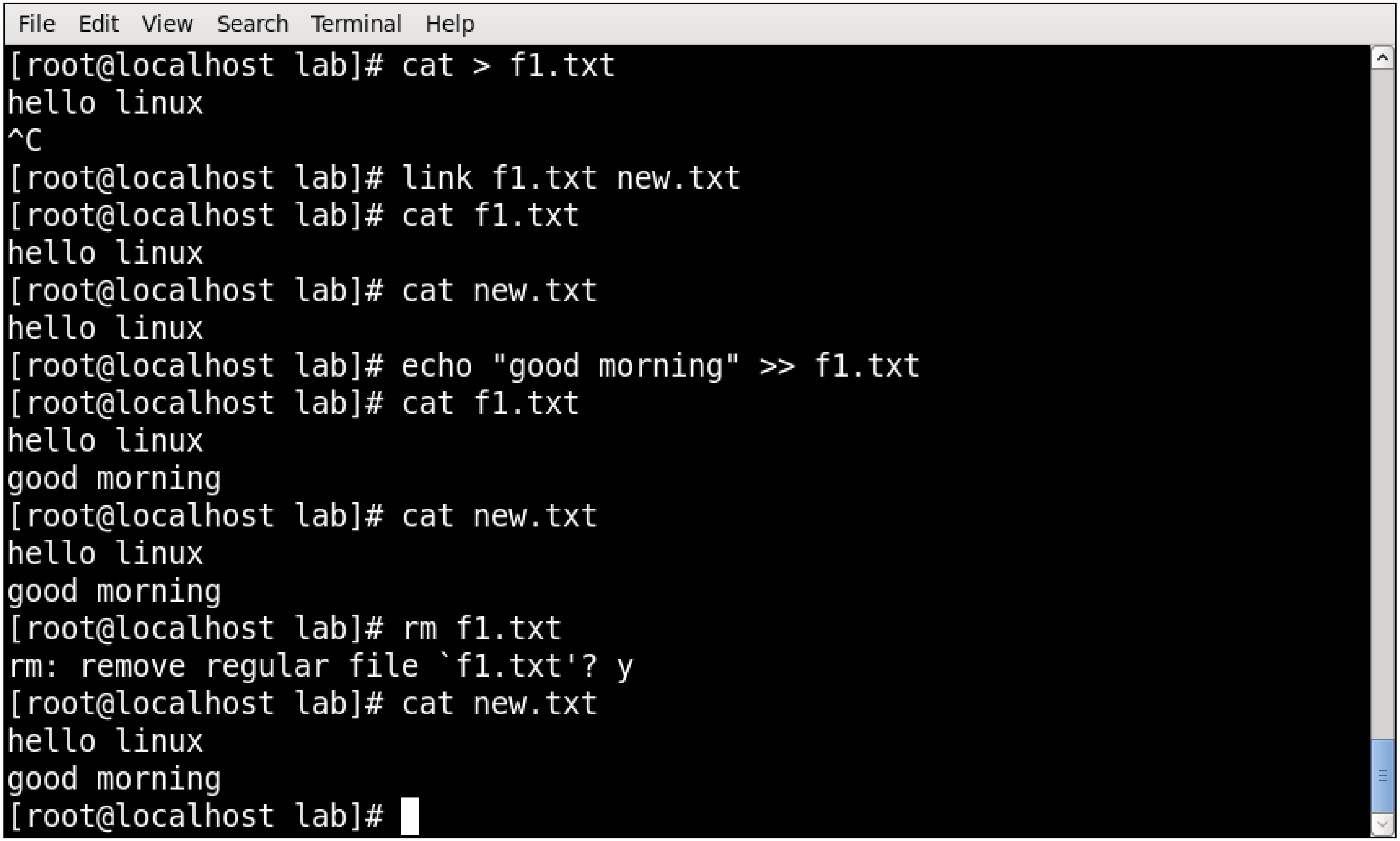
- **ln** creates links between files.
- ln creates hard links by default, or symbolic links if the -s (--symbolic) option is specified. When creating hard links, each TARGET must exist.

- **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

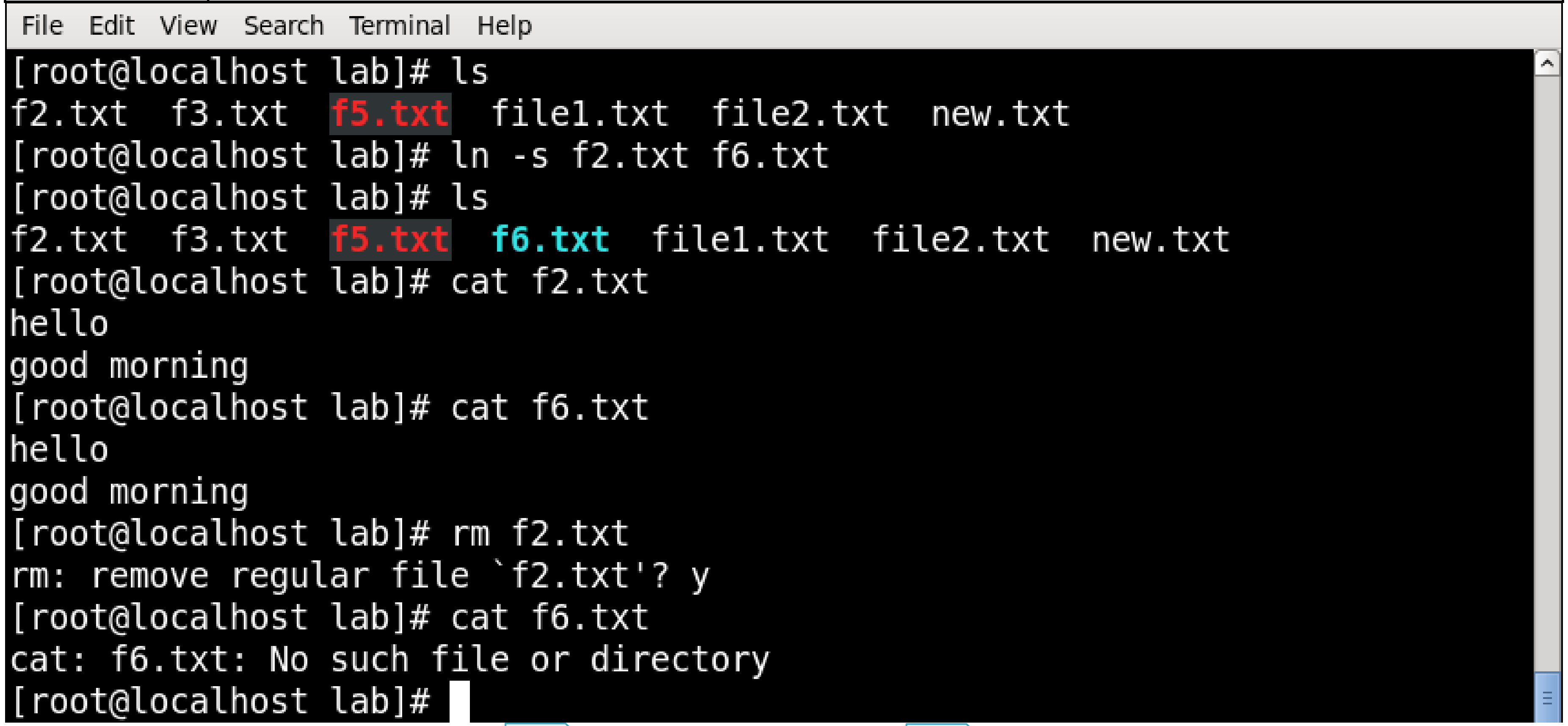
ln Command Example



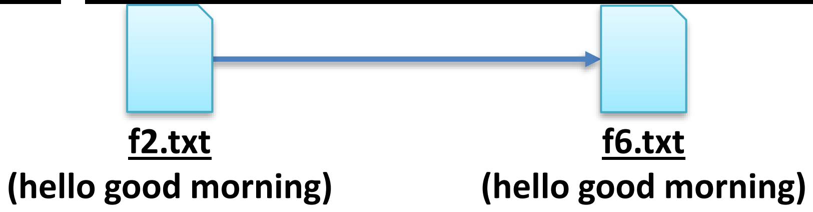
```
[root@localhost lab]# cat > f1.txt
hello linux
^C
[root@localhost lab]# link f1.txt new.txt
[root@localhost lab]# cat f1.txt
hello linux
[root@localhost lab]# cat new.txt
hello linux
[root@localhost lab]# echo "good morning" >> f1.txt
[root@localhost lab]# cat f1.txt
hello linux
good morning
[root@localhost 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]#
```



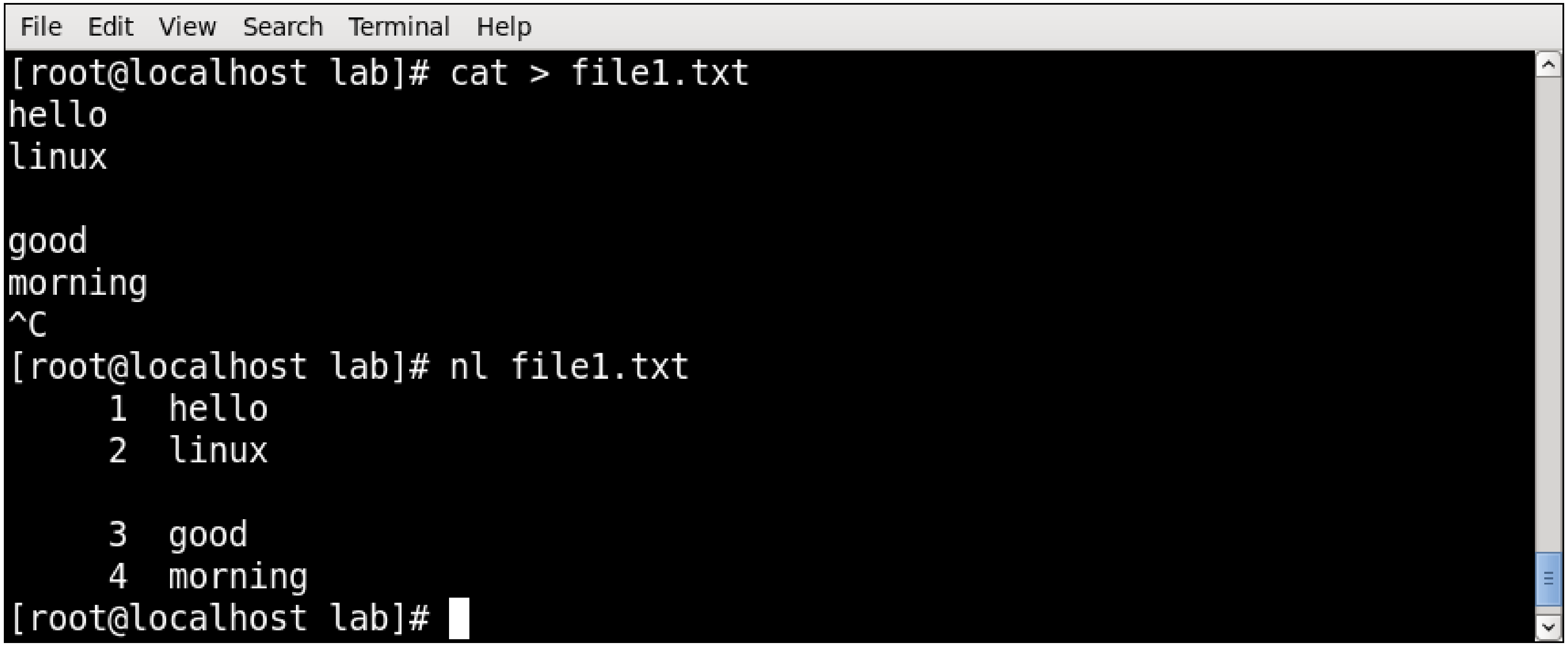
nl Command

- **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



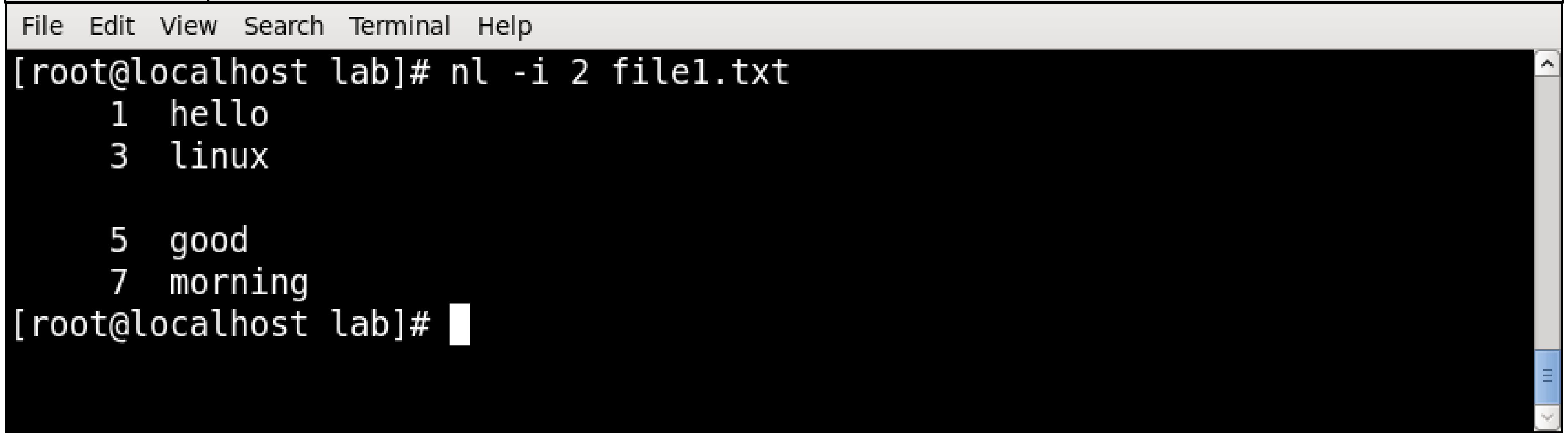
```
[root@localhost lab]# cat > file1.txt
hello
linux

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**

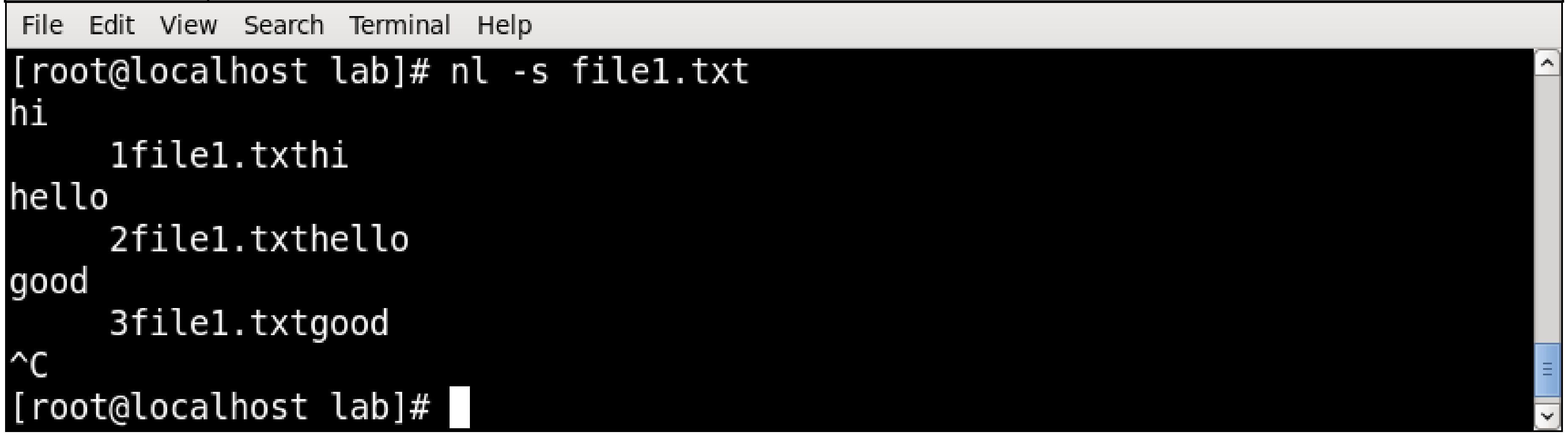


```
[root@localhost lab]# nl -i 2 file1.txt
     1  hello
     3  linux

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

nl Command Example

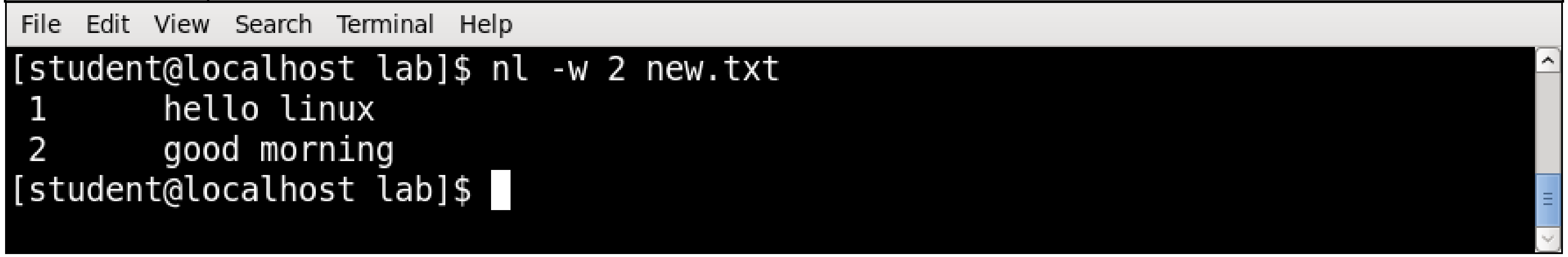
nl -s	**Add STRING after (possible) line number**



```
[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

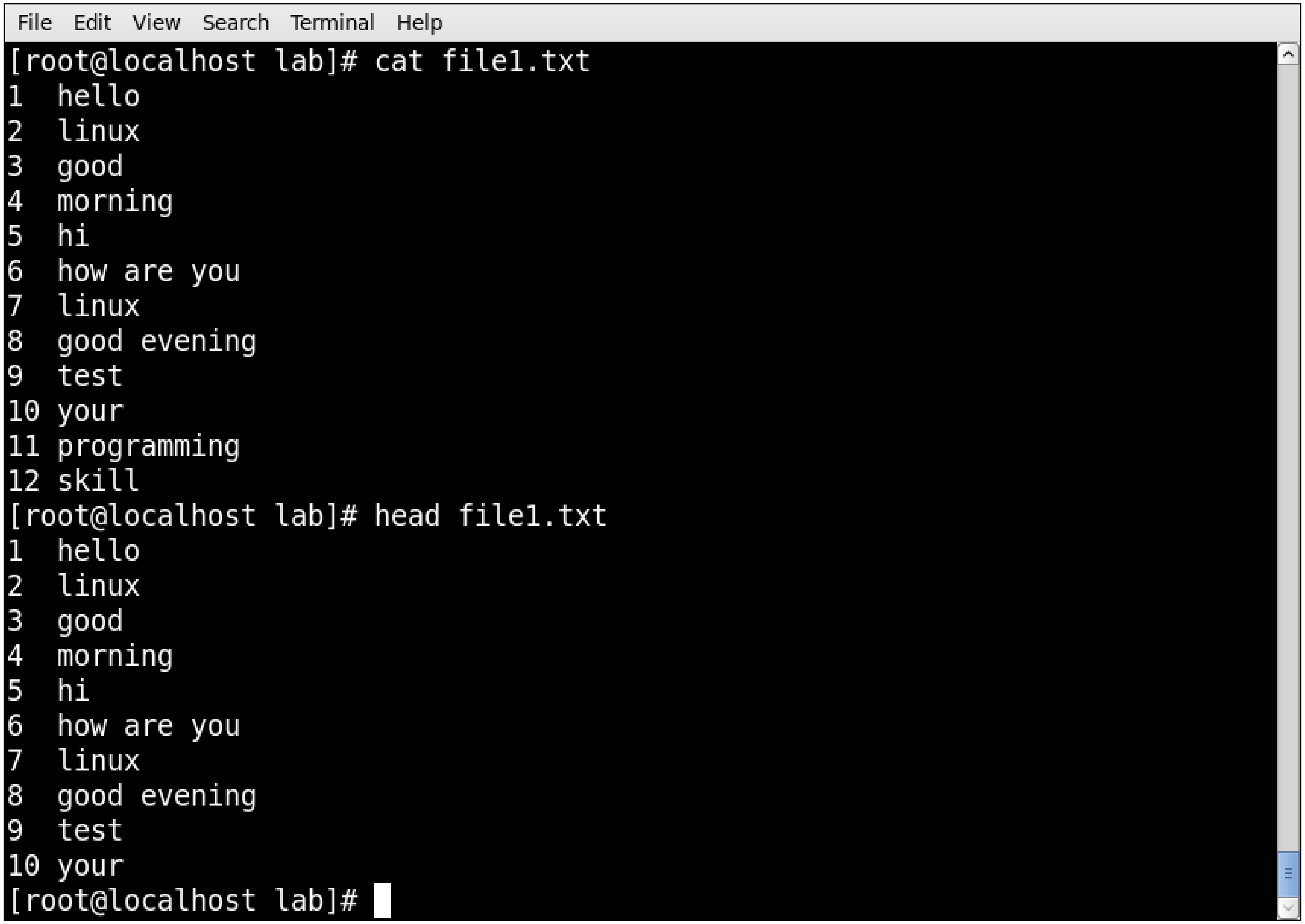
- **head** makes it easy to output the first part (10 lines by default) of files.
- **Syntax :**

  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

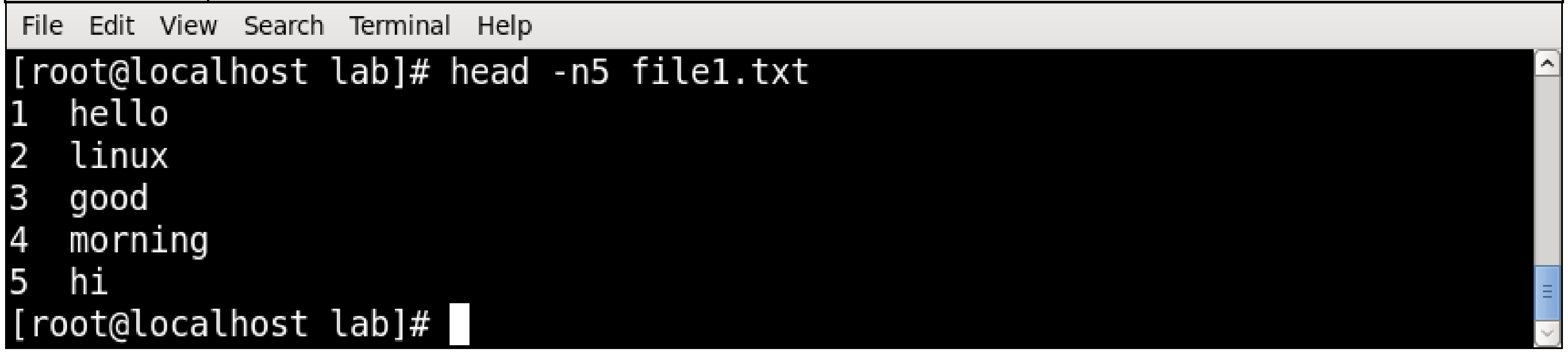
head Command Example



```
[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

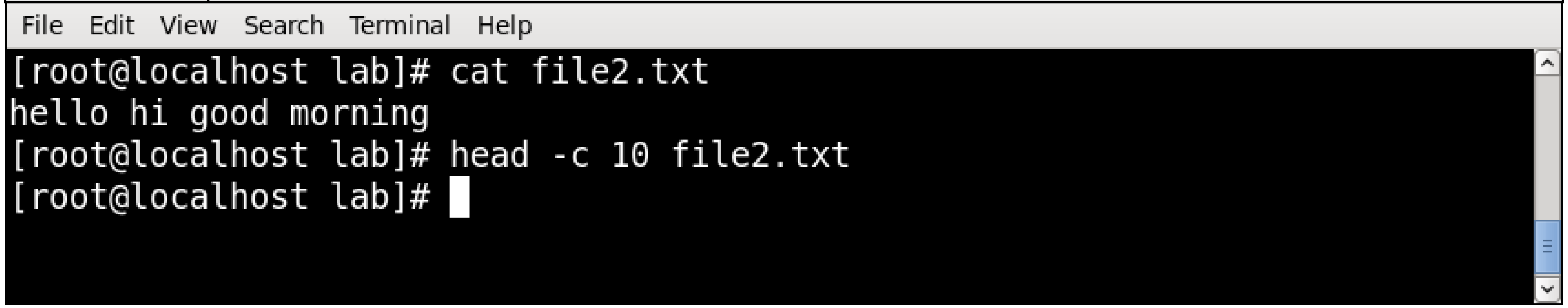
head -n	**Print the first n lines instead of the first 10; with the leading '-', print all but the last n lines of each file**



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

head Command Example

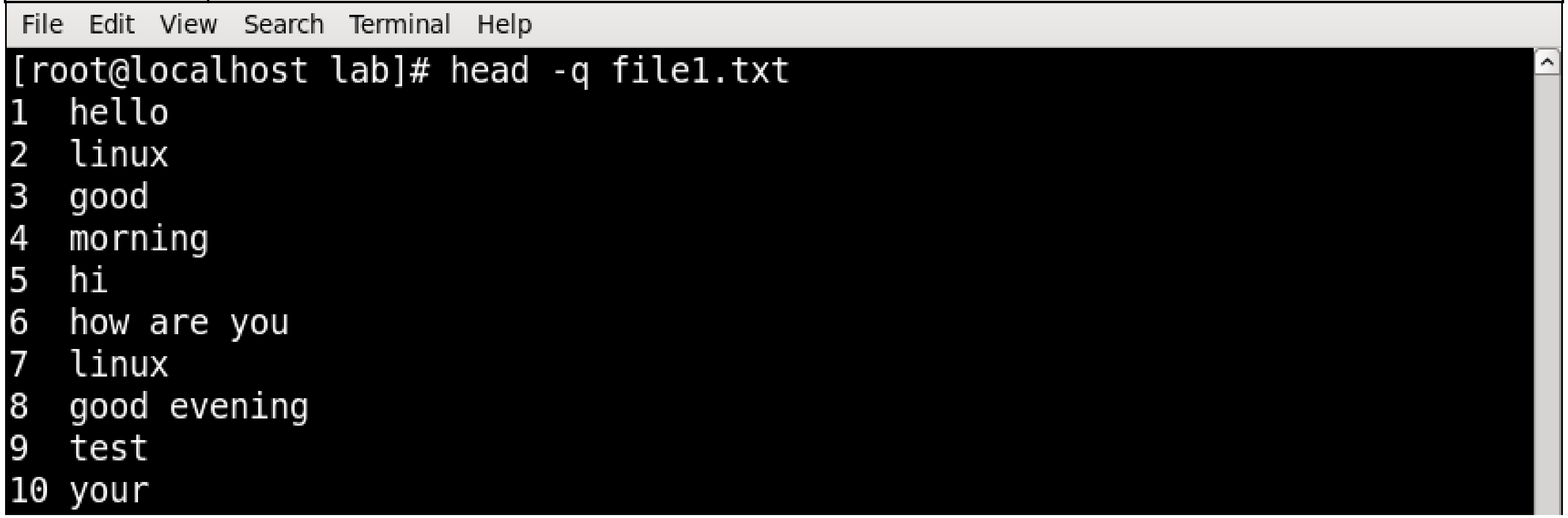
head -c	**Print the first n bytes of each file; with a leading '-', print all but the last n bytes of each file**



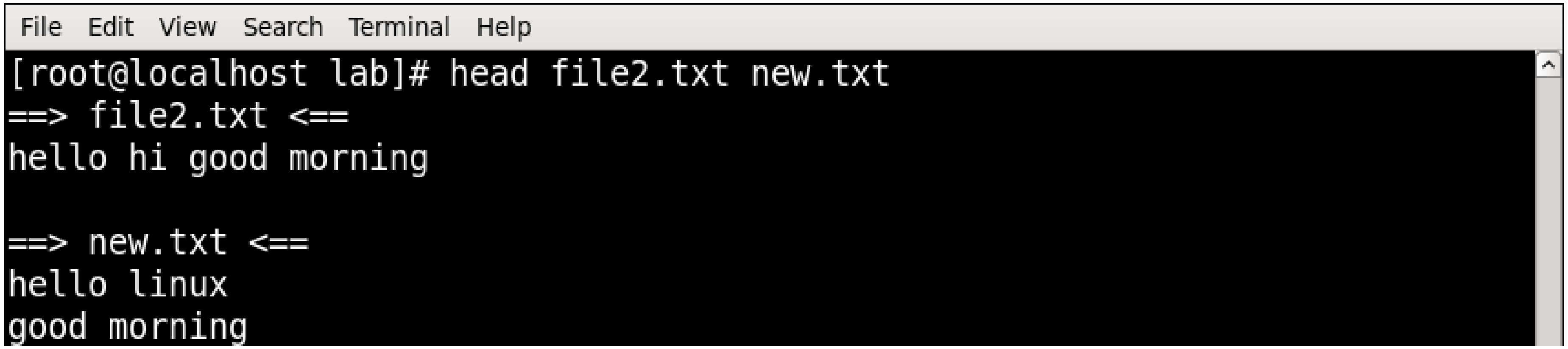
```
[root@localhost lab]# cat file2.txt
hello hi good morning
[root@localhost lab]# head -c 10 file2.txt
[root@localhost lab]# 
```

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

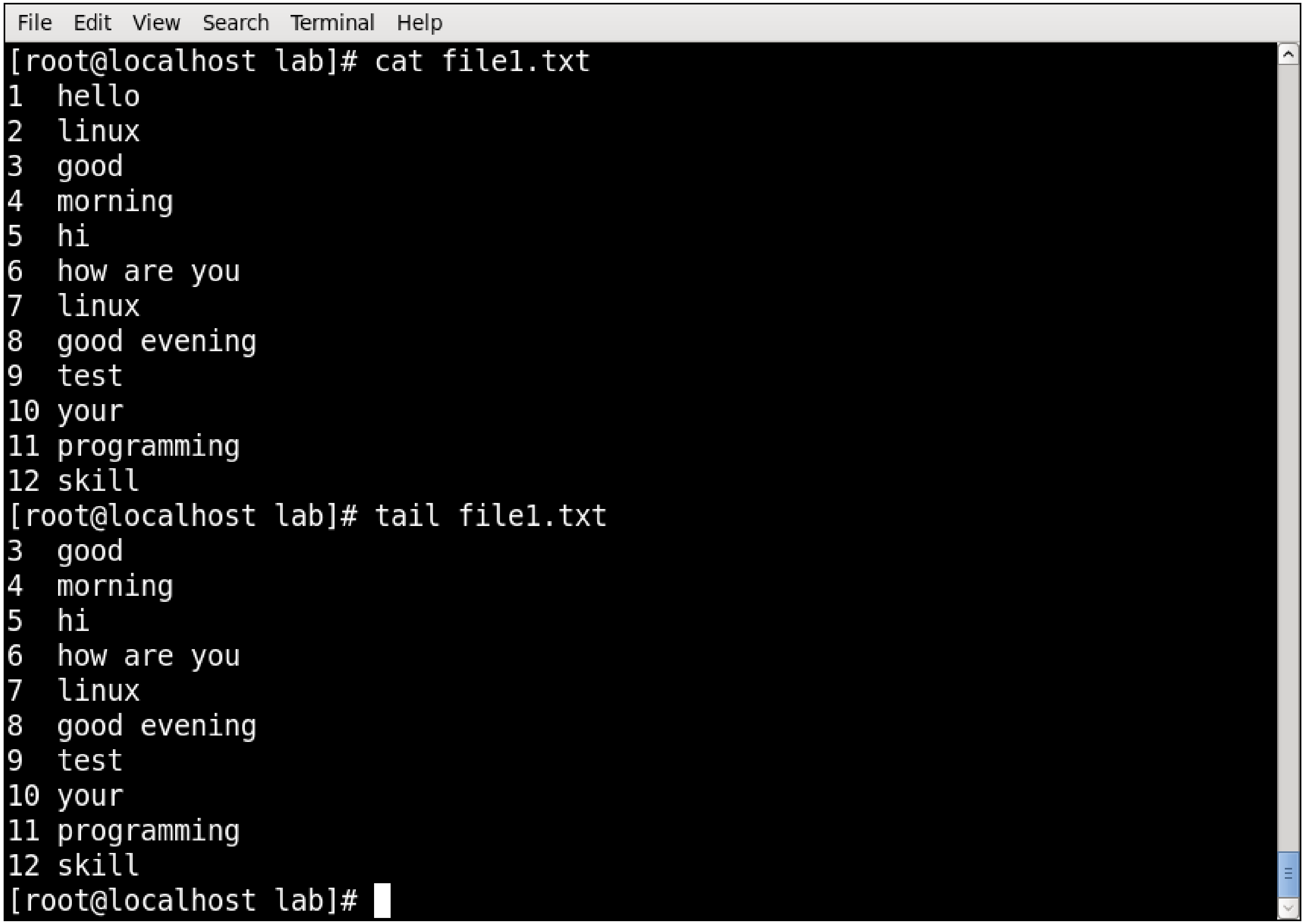
- **tail** is a command which prints the last few number of lines (10 lines by default) of a certain file, then terminates.

- **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

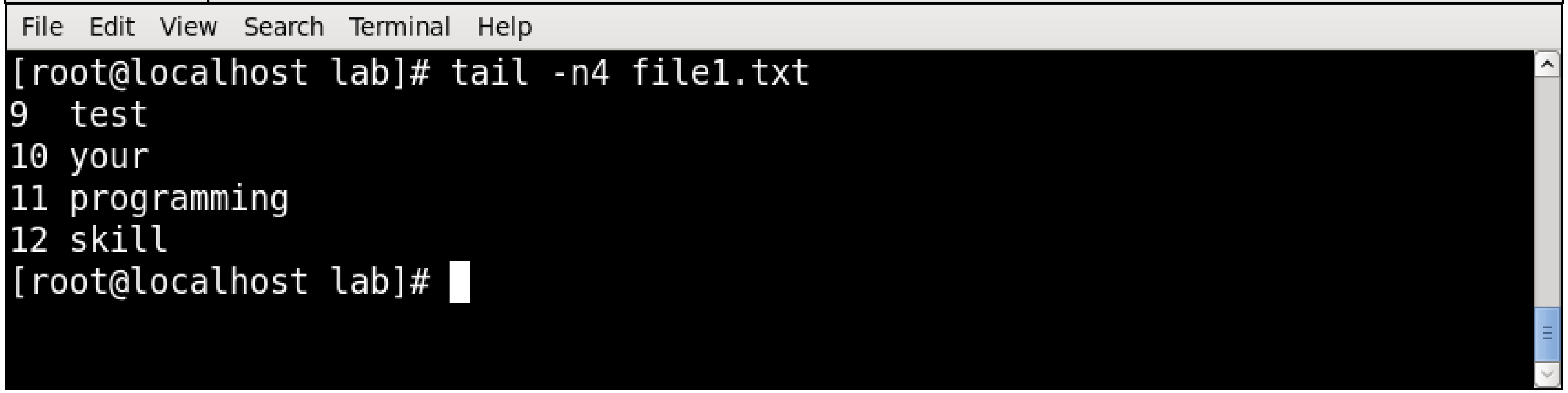
tail Command Example



```
[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

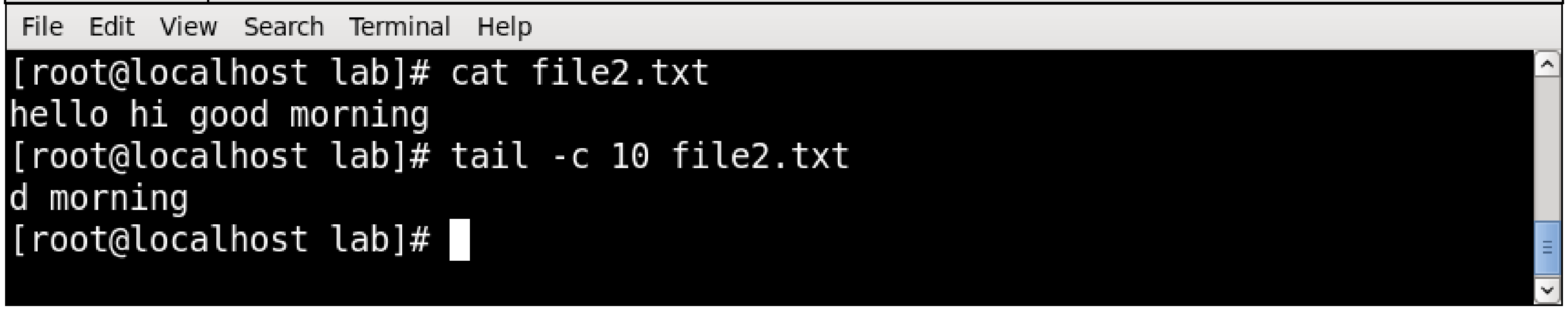
tail -n	**Output the last num lines, instead of the default (10)**



```
[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]# 
```

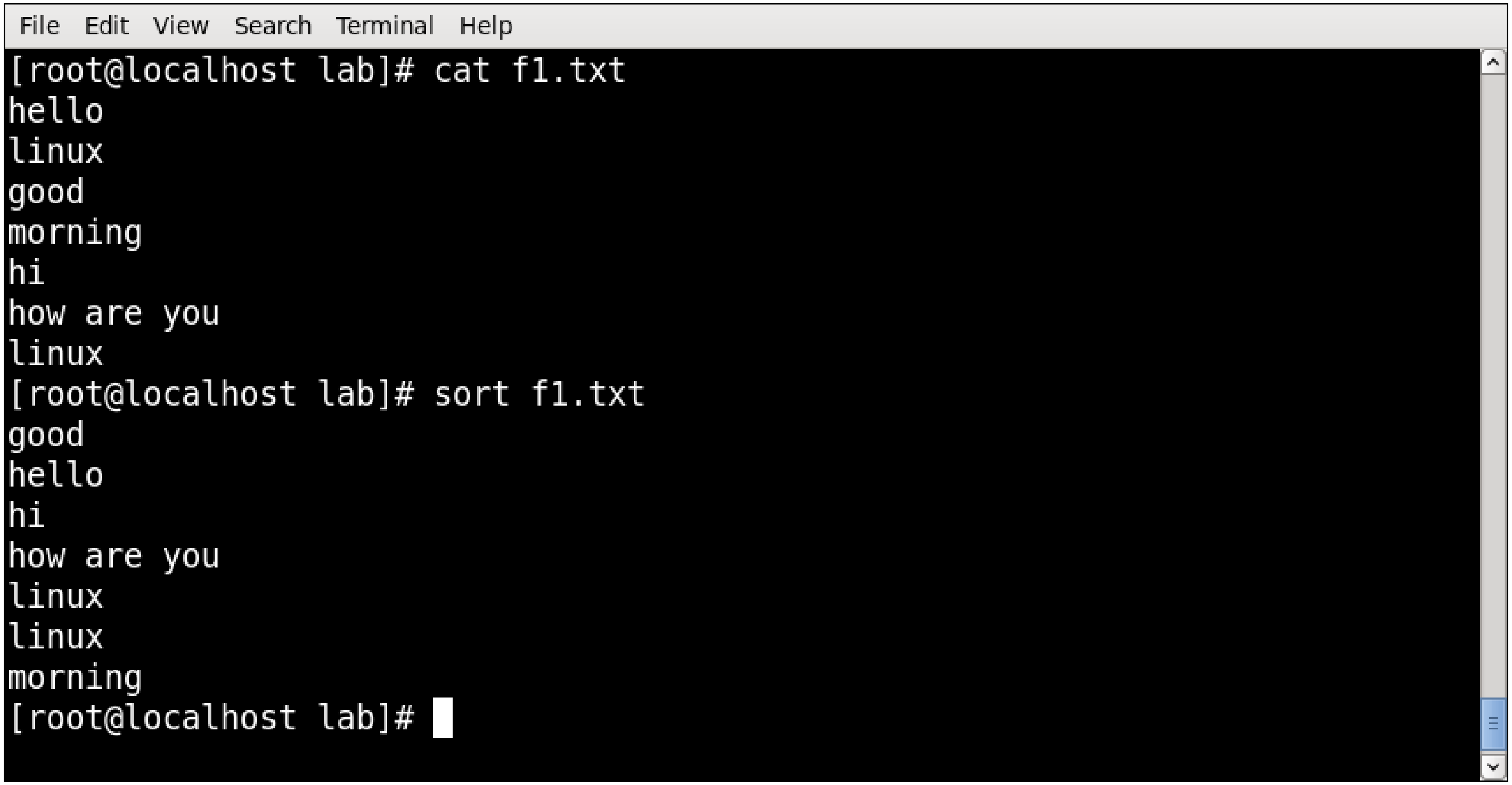
sort Command

- **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

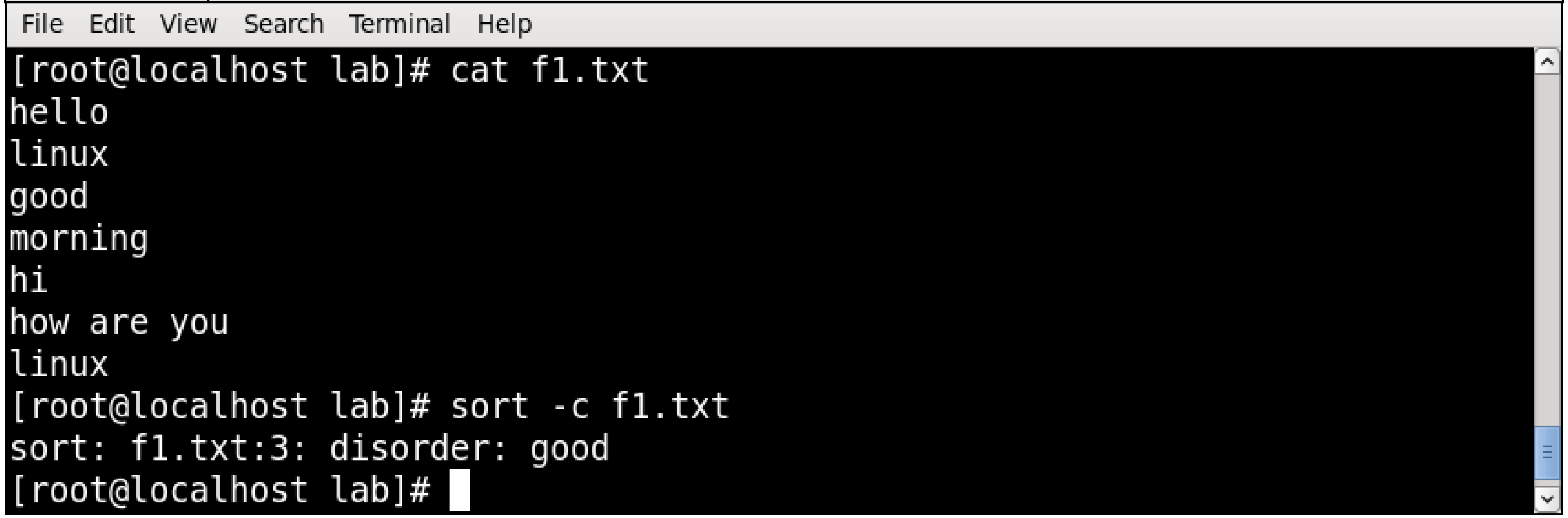
sort Command Example



```
[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

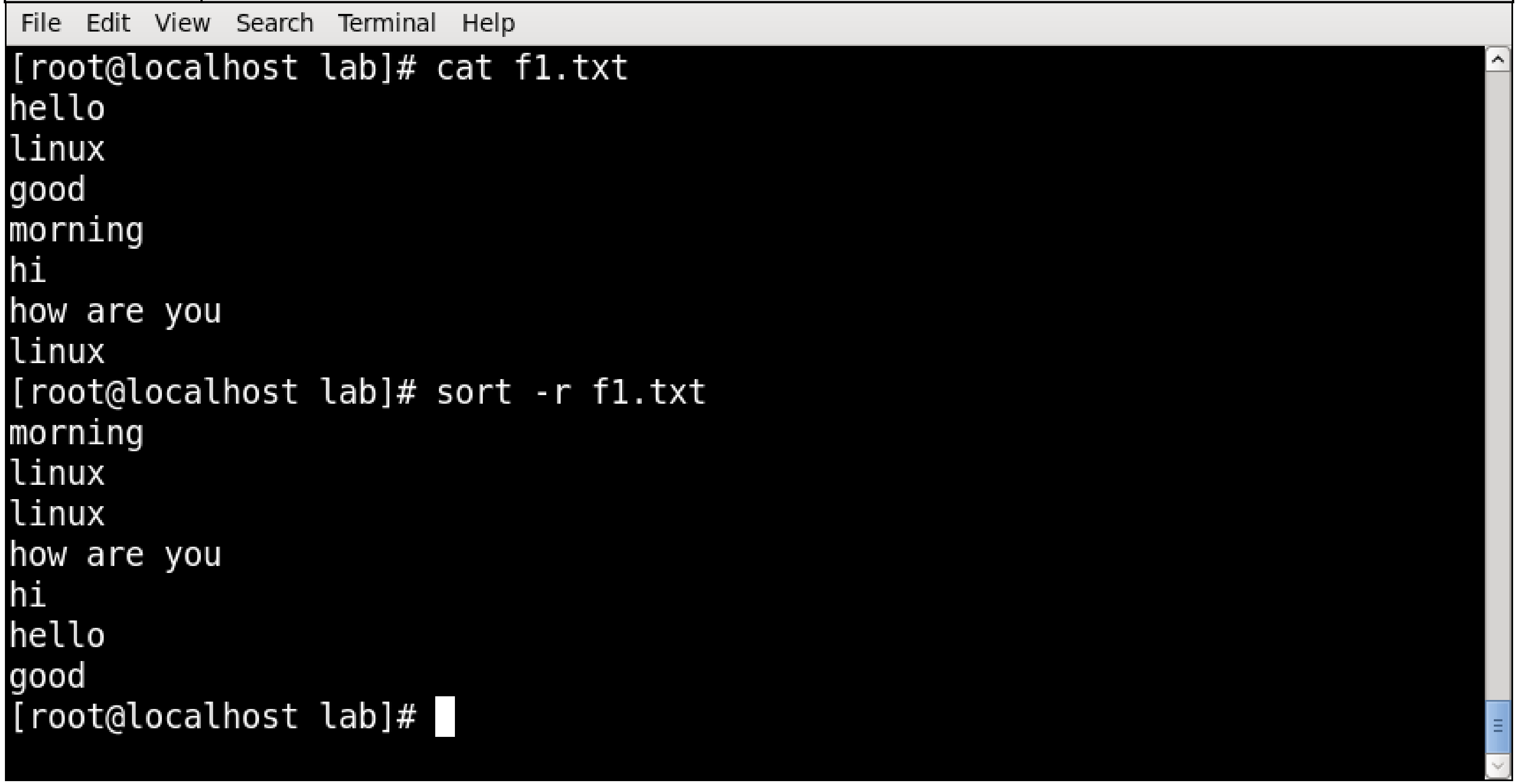
sort -c	**To check if the file given is already sorted or not**



```
[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

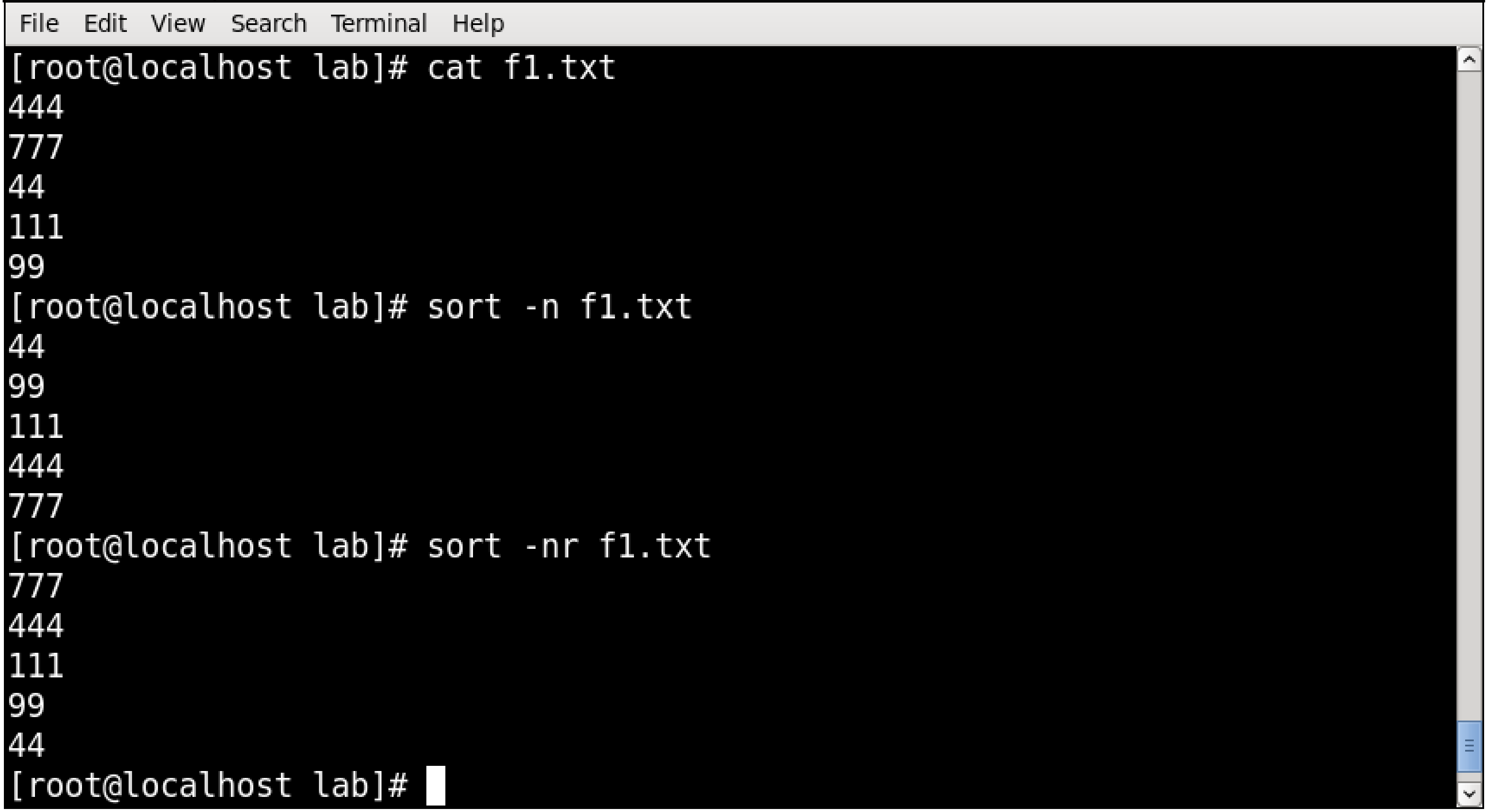
sort -r	**Reverse the result of comparisons**



```
[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

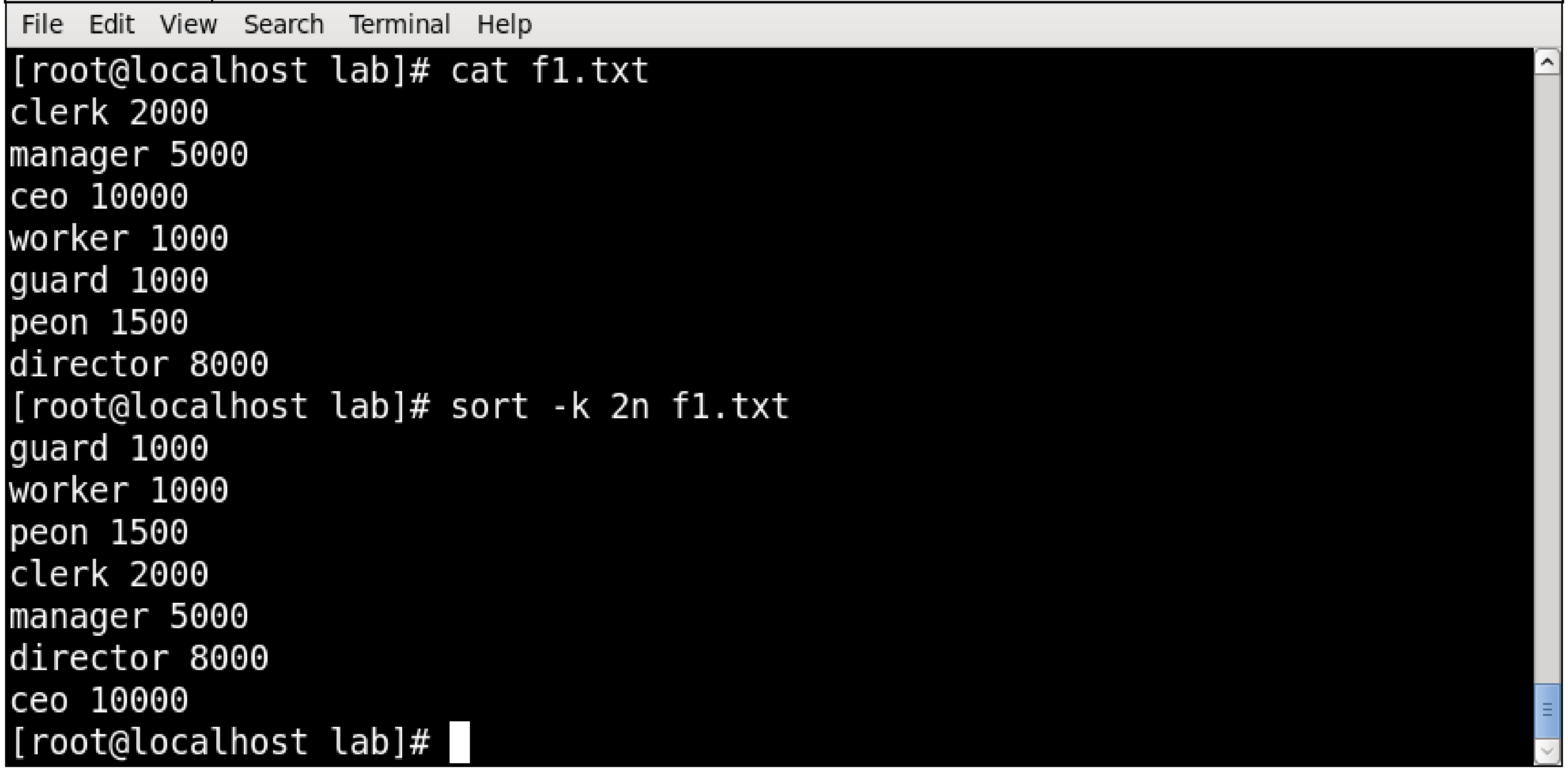
sort -n	**Compare according to string numerical value**
sort -nr	**To sort a file with numeric data in reverse order**



```
[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]#
```

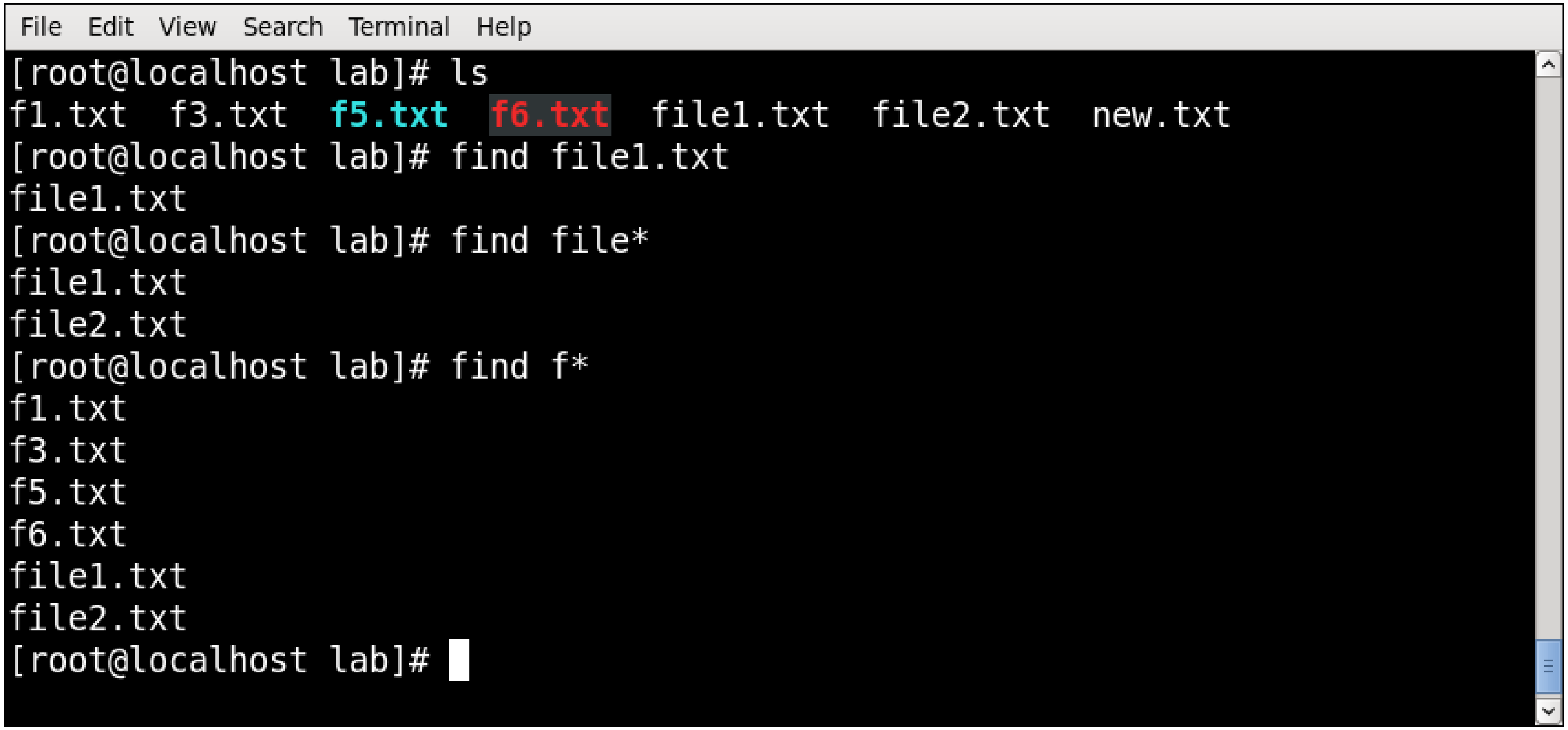
find Command

- **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’

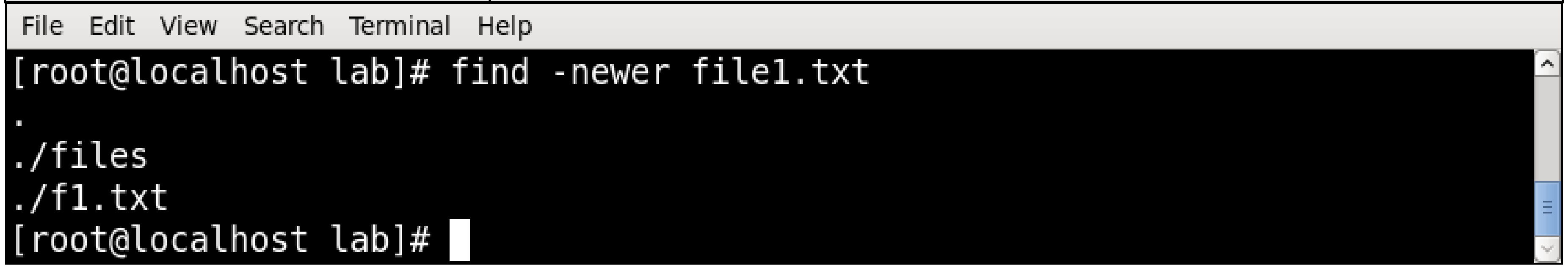
find Command Example



```
[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

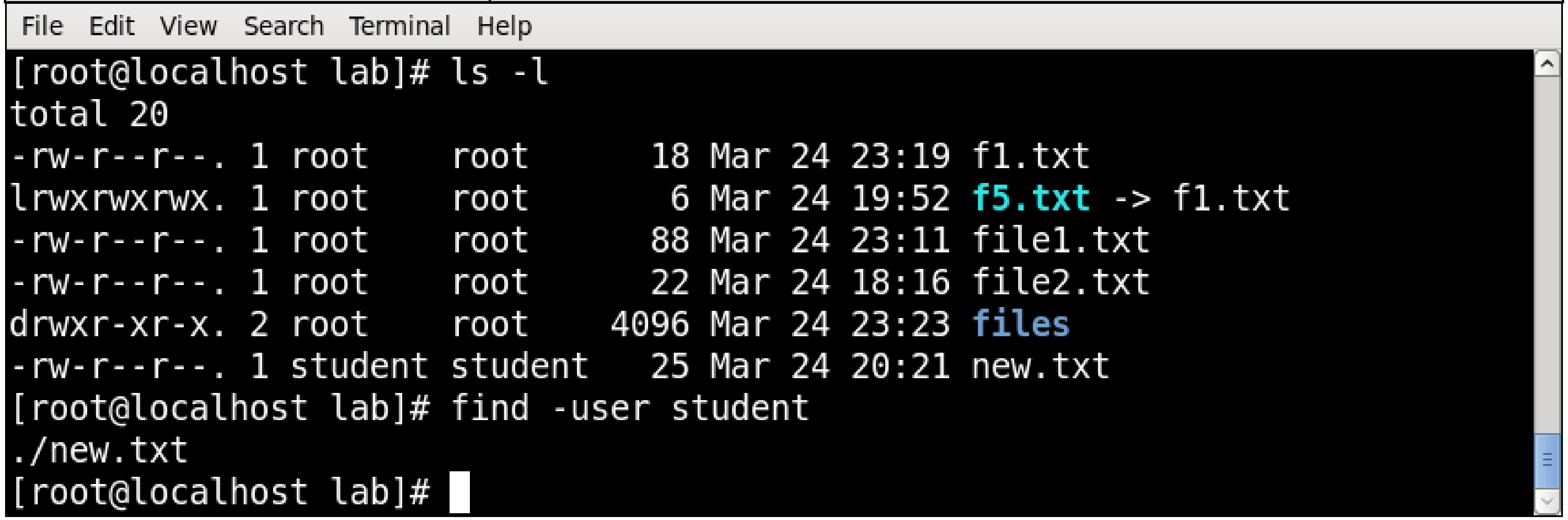
find -newer filename	**Search for files that were modified/created after 'filename'**



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

find Command Example

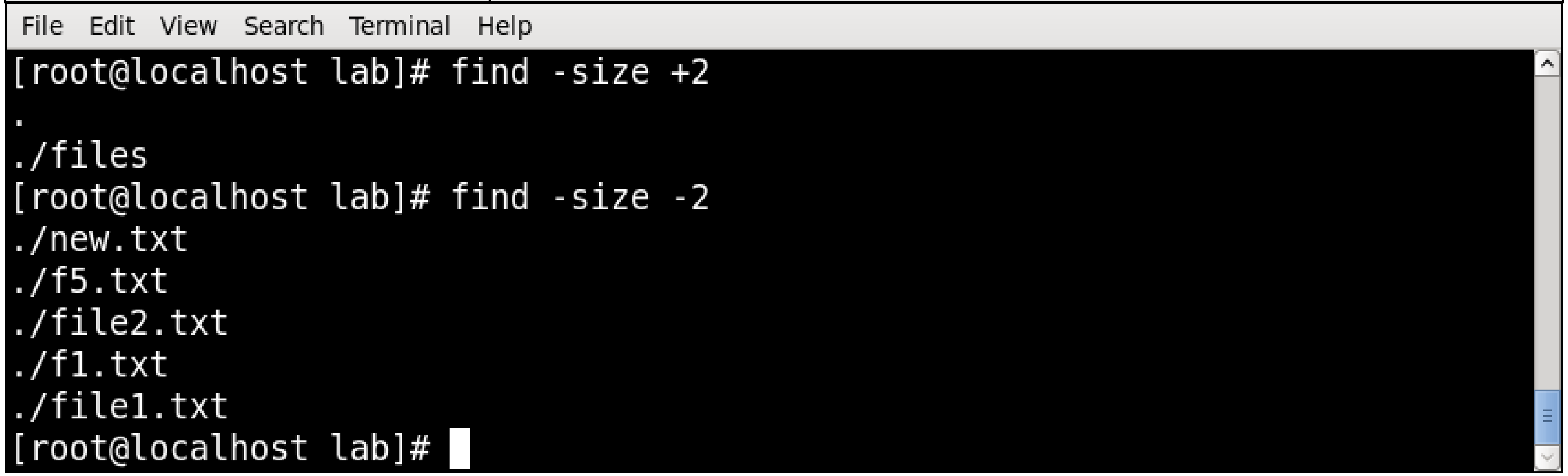
find -user name	**Search for files owned by user name or ID ‘name’**



```
[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

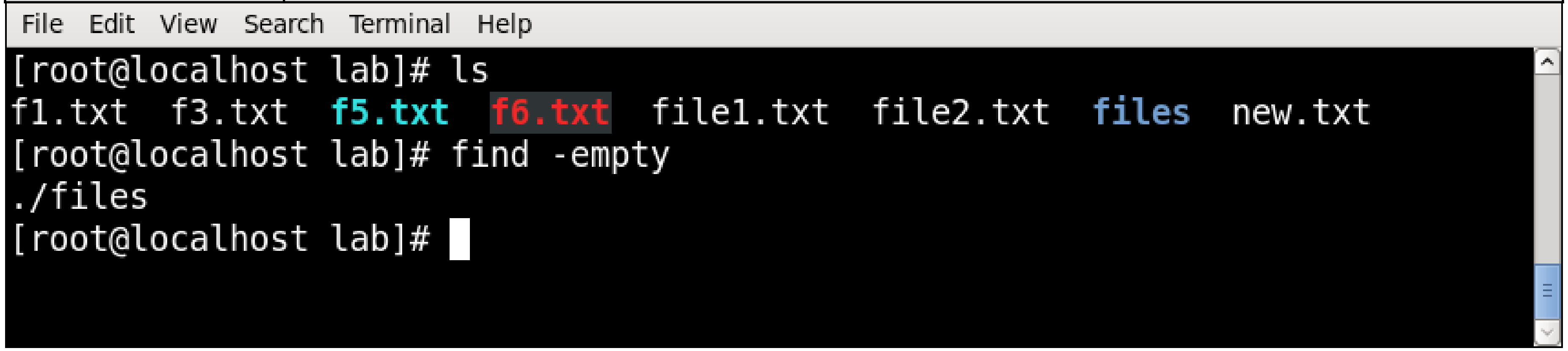
find -size +N/-N	**Search for files of ‘N’ blocks; ‘N’ followed by ‘c’ can be used to measure size in characters**



```
[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

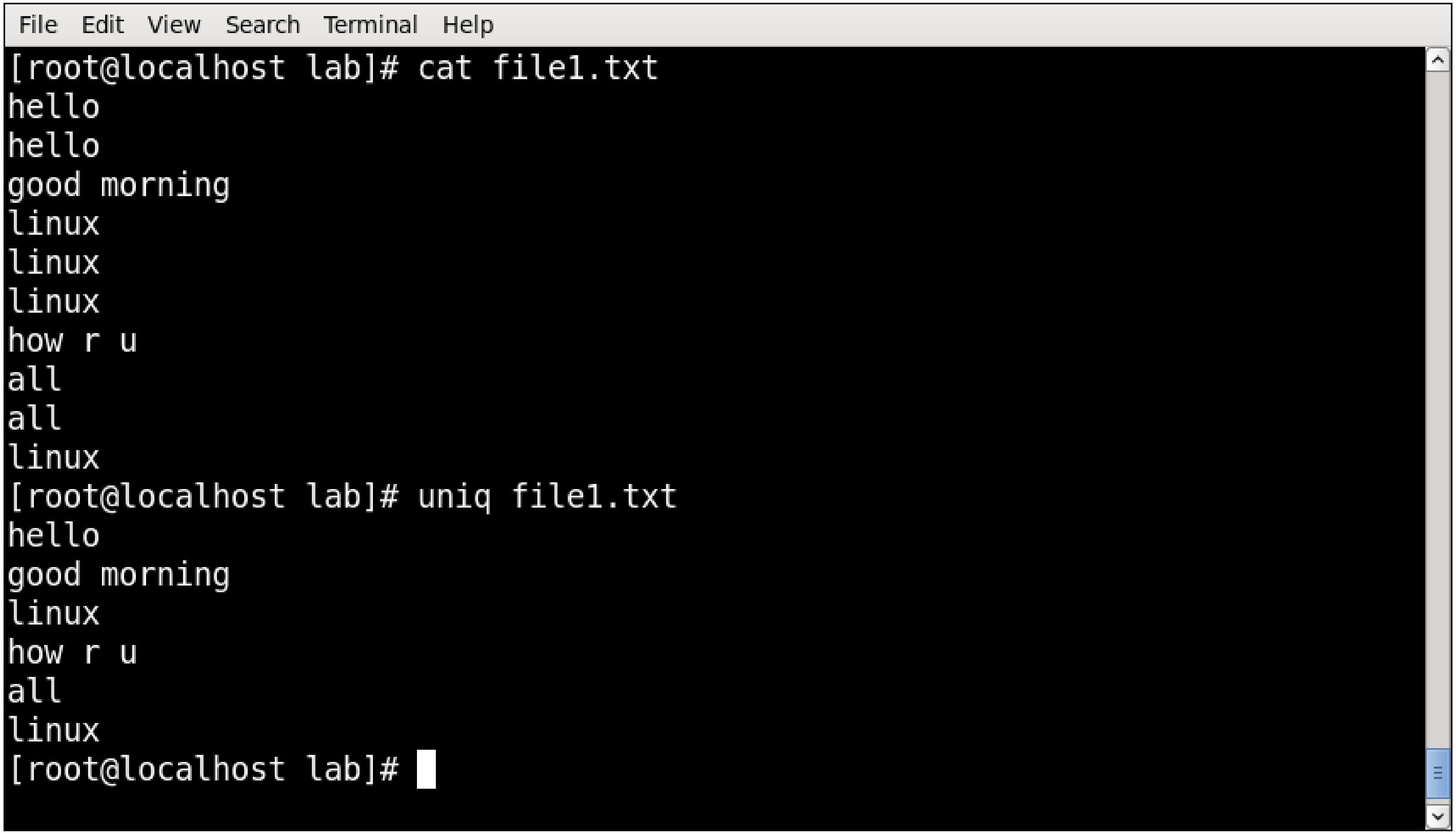
- **uniq** reports or filters out repeated lines in a file.
- It can remove duplicates, show a count of occurrences, show only repeated lines, ignore certain characters and compare on specific fields.

- **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

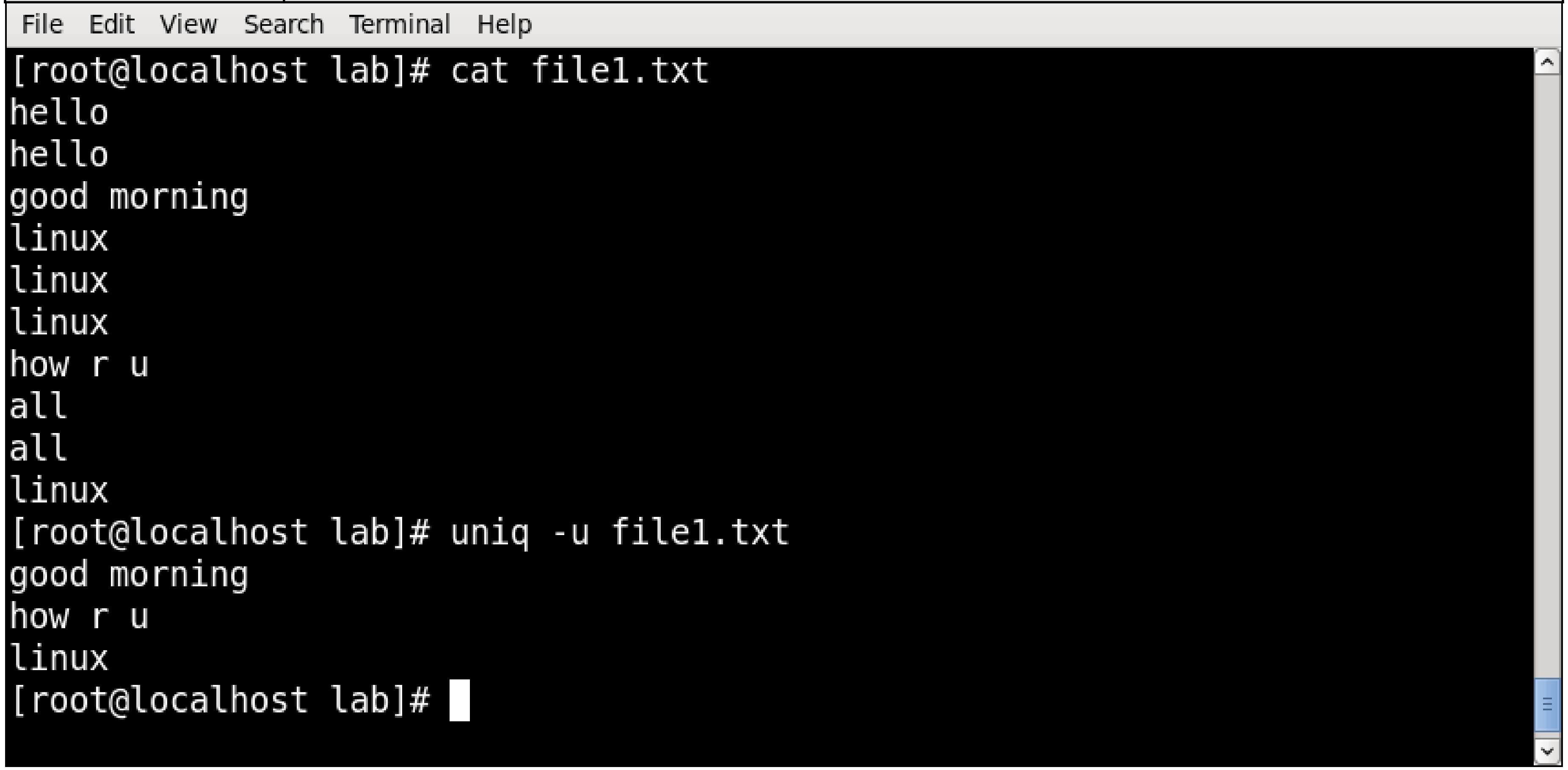
uniq Command Example



```
[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

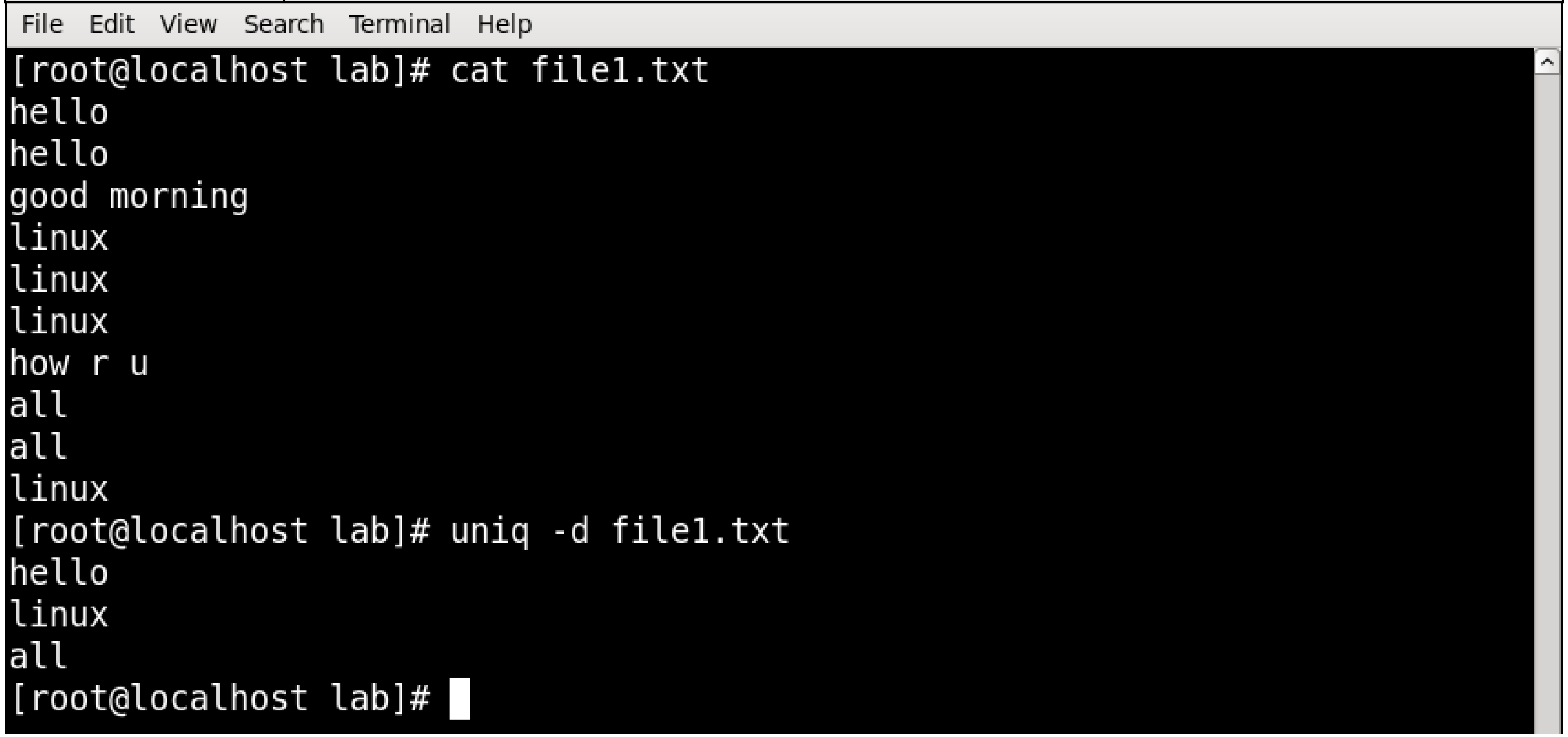
uniq -u	**Prints only unique lines**



```
[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]# 
```

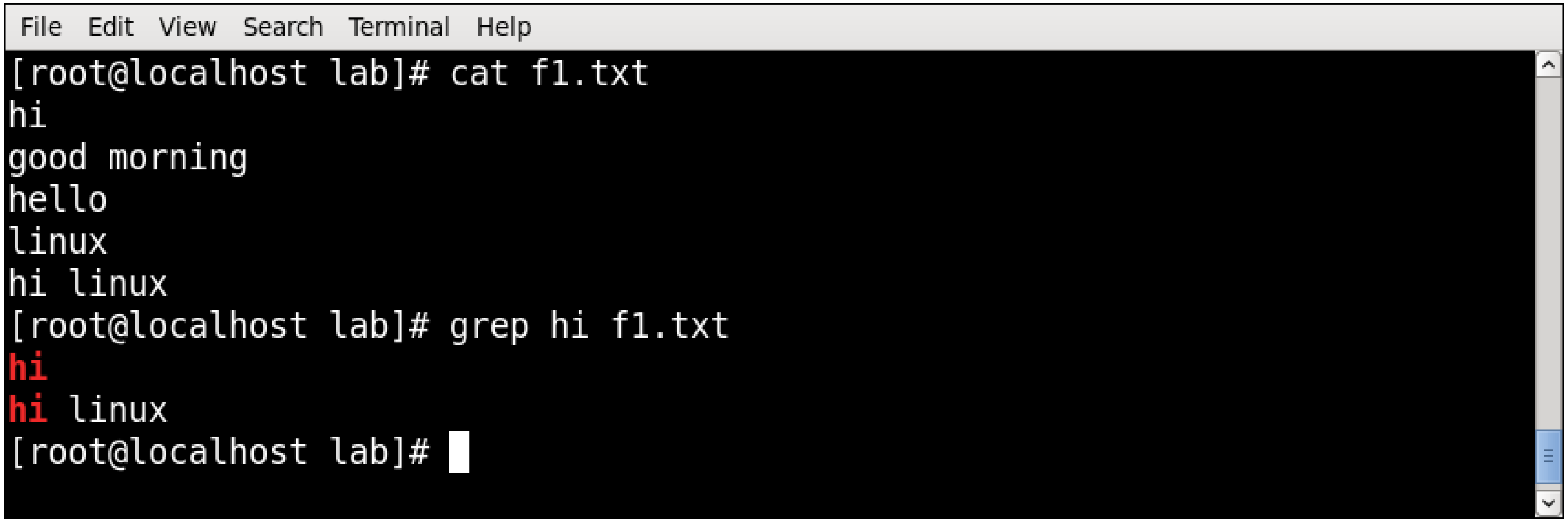
grep Command

- 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

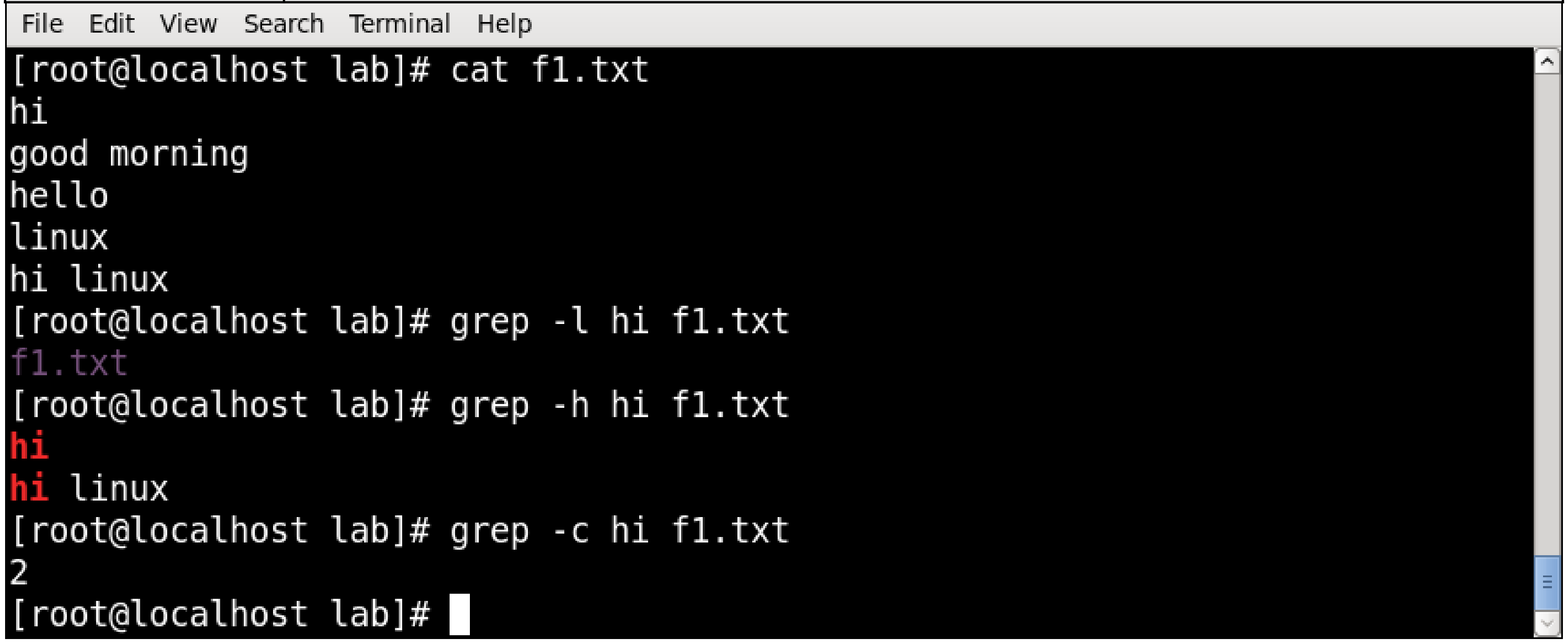
grep Command Example



```
[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

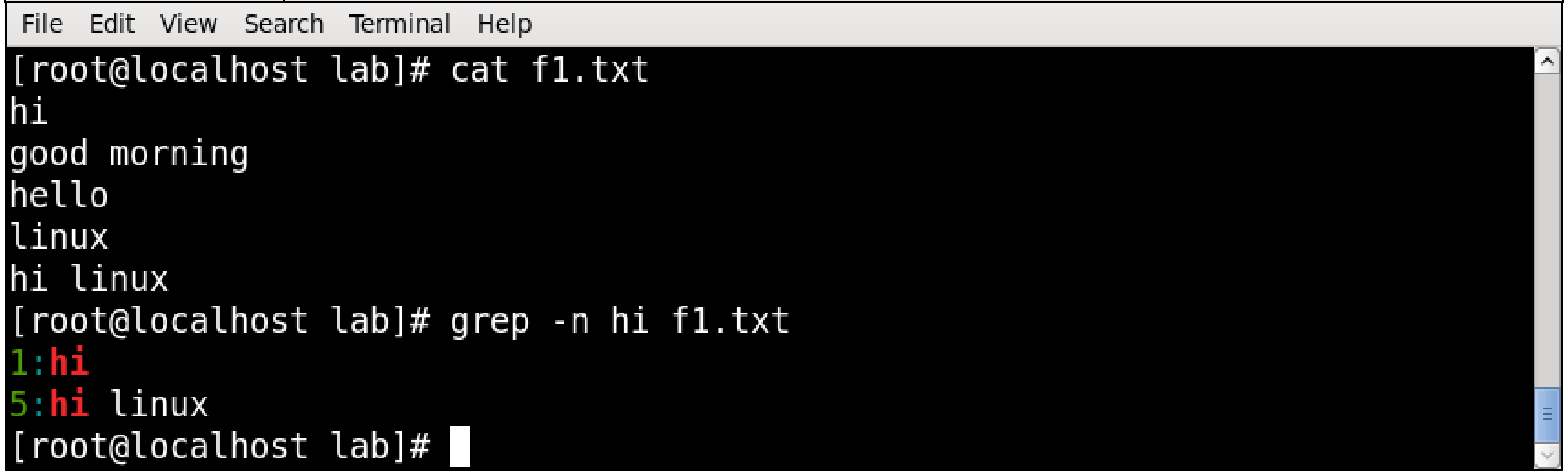
grep -c	Prints only a count of the lines that match a pattern
grep -h	Display the matched lines, but do not display the filenames
grep -l	Displays list of a filenames only



```
[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

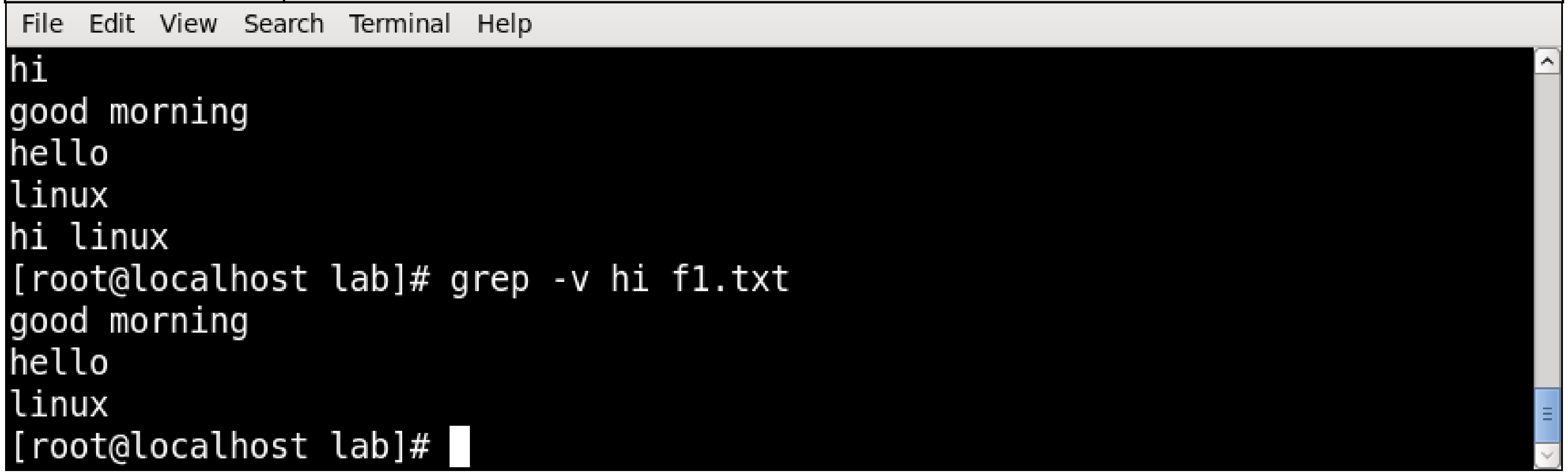
grep -n	**Display the matched lines and their line numbers**



```
[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

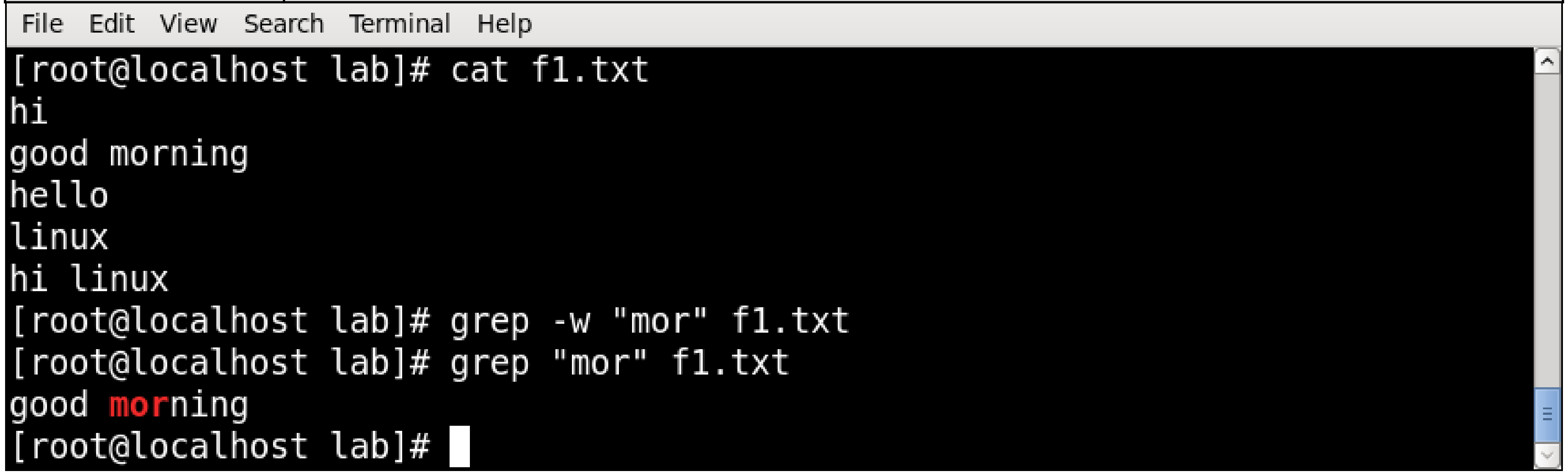
grep -v	**This prints out all the lines that do not matches the pattern**



```
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

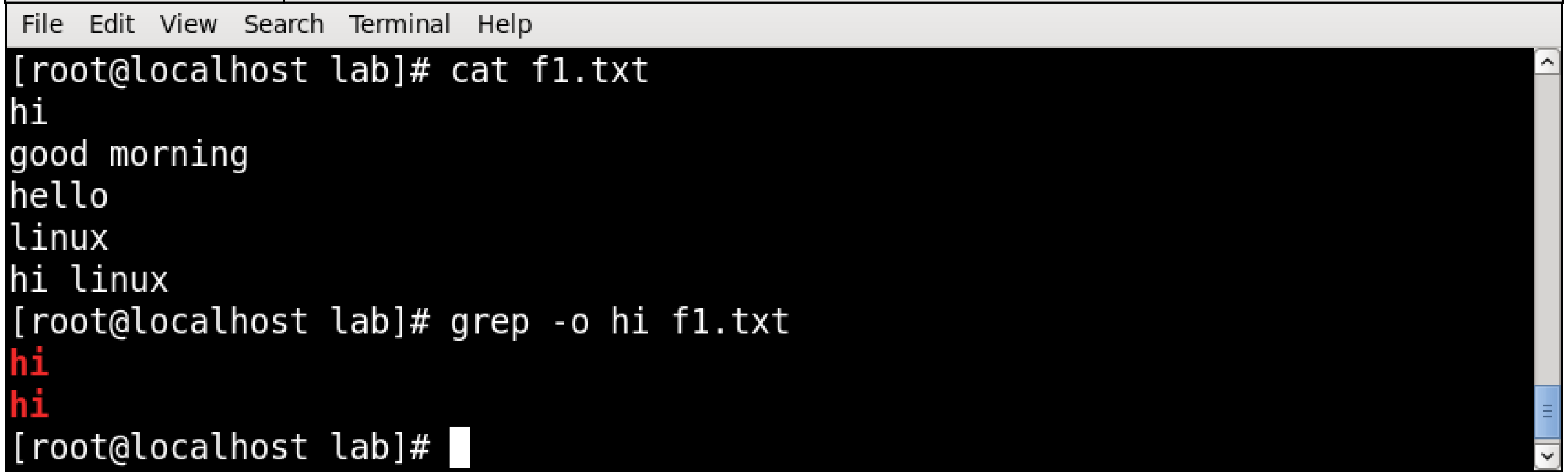
grep -w	**Match whole word**



```
[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]#
```

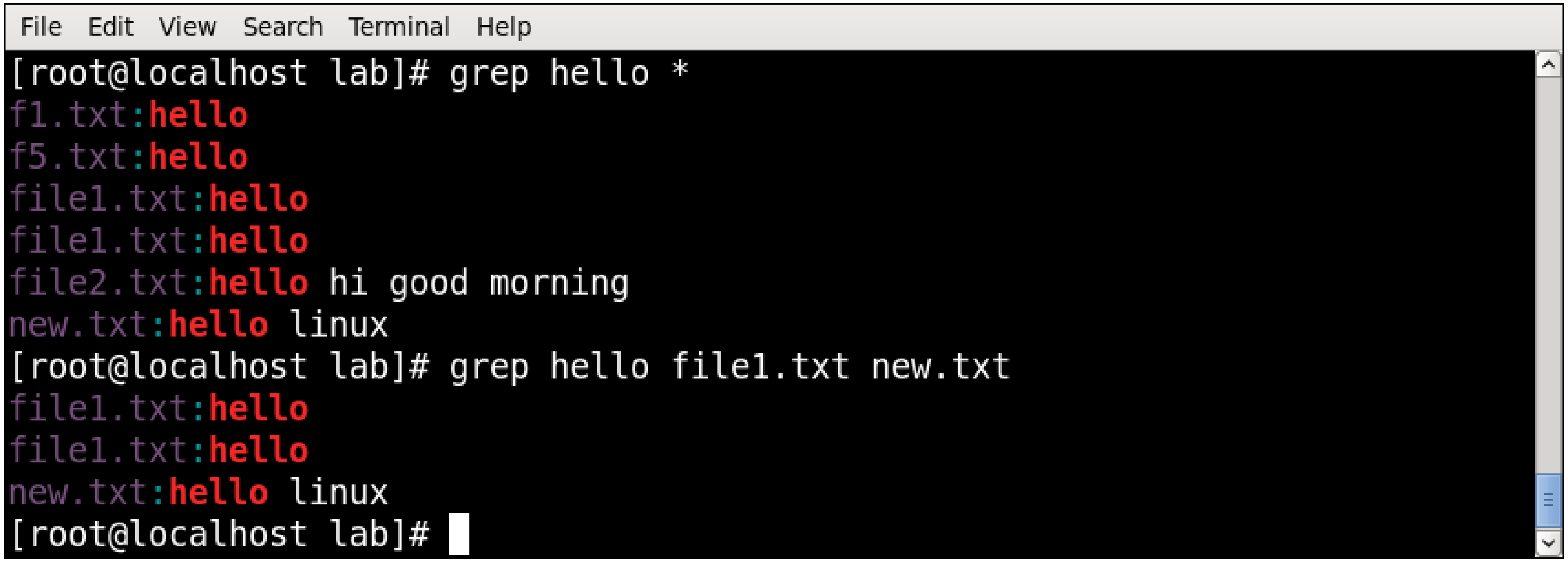
grep Command Example

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



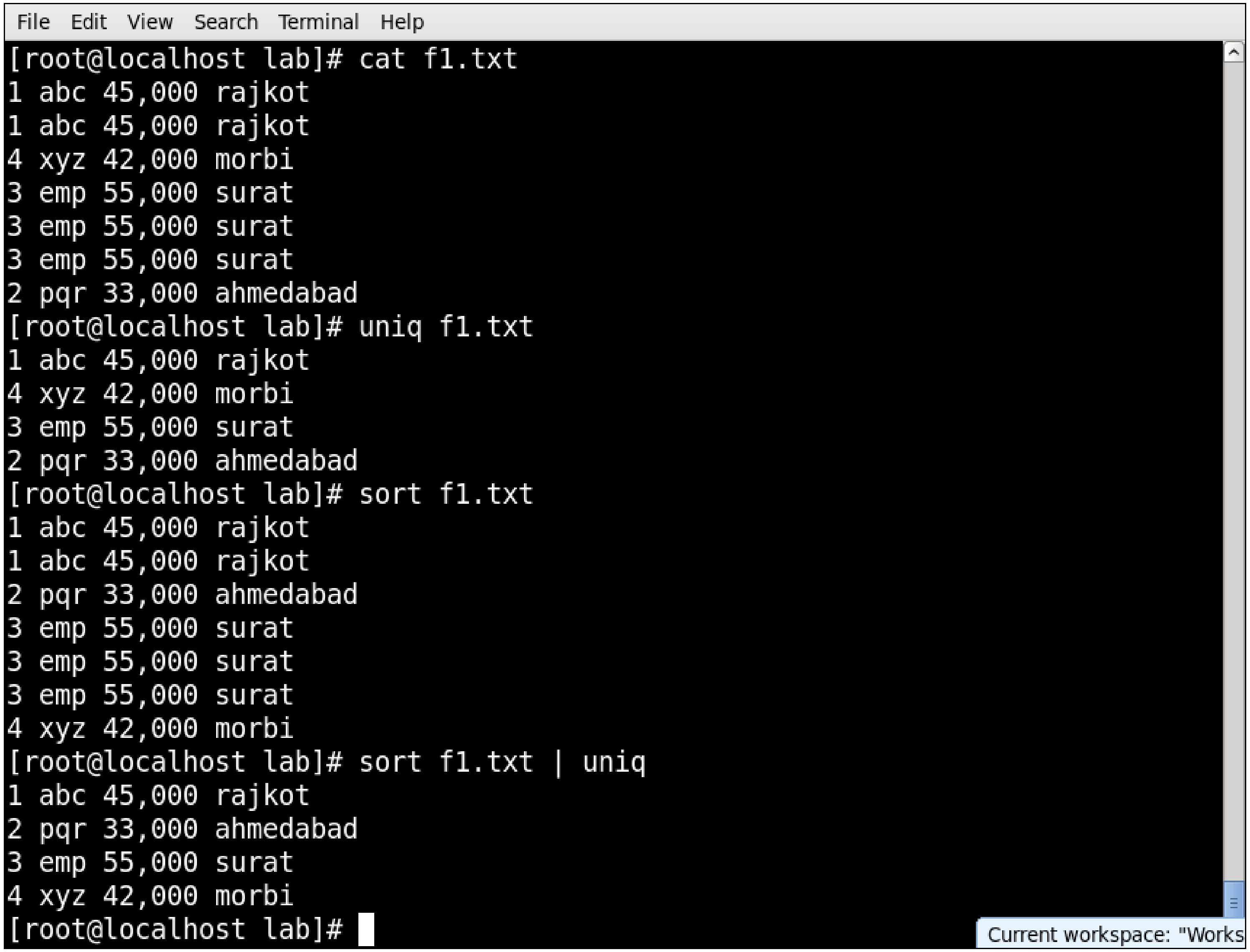
```
[root@localhost lab]# grep hello *
f1.txt:hello
f5.txt:hello
file1.txt:hello
file1.txt:hello
file2.txt:hello hi good morning
new.txt:hello linux
[root@localhost lab]# grep hello file1.txt new.txt
file1.txt:hello
file1.txt:hello
new.txt:hello linux
[root@localhost lab]# 
```

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 :**

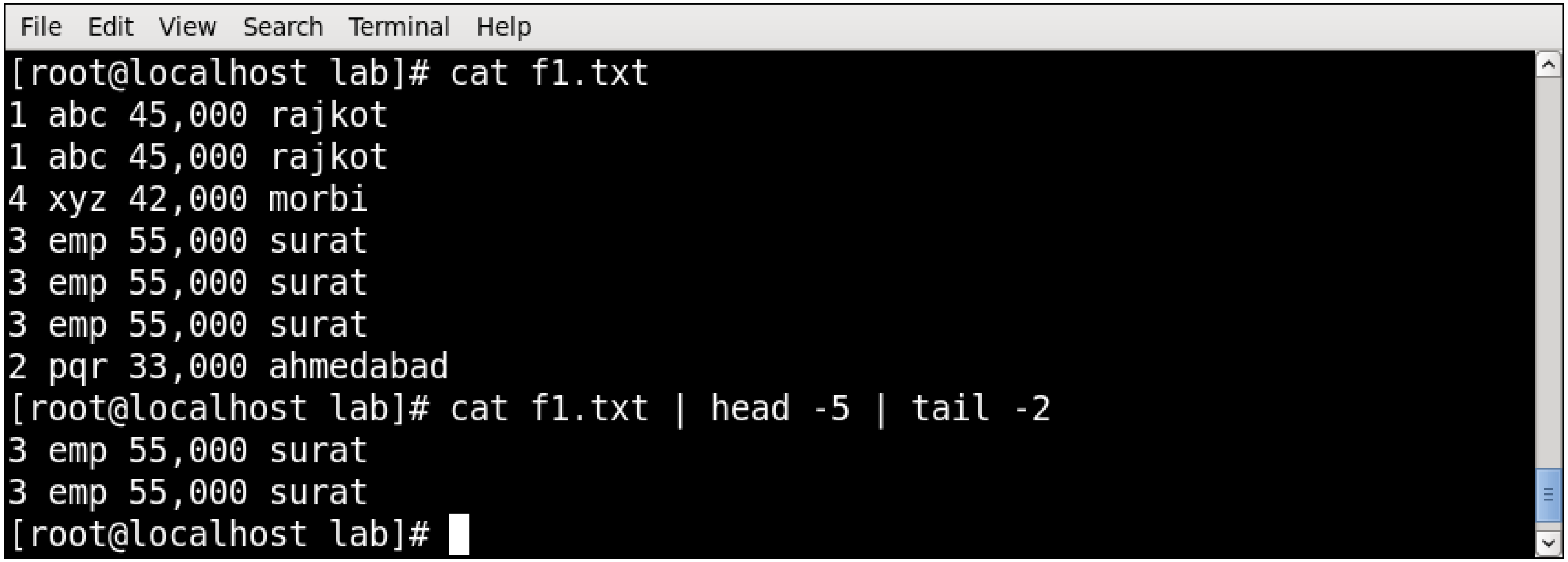
  command_1 | command_2 | command_3 | .... | command_N...

pipe Command Example


File Edit View Search Terminal Help
Current workspace: "Works

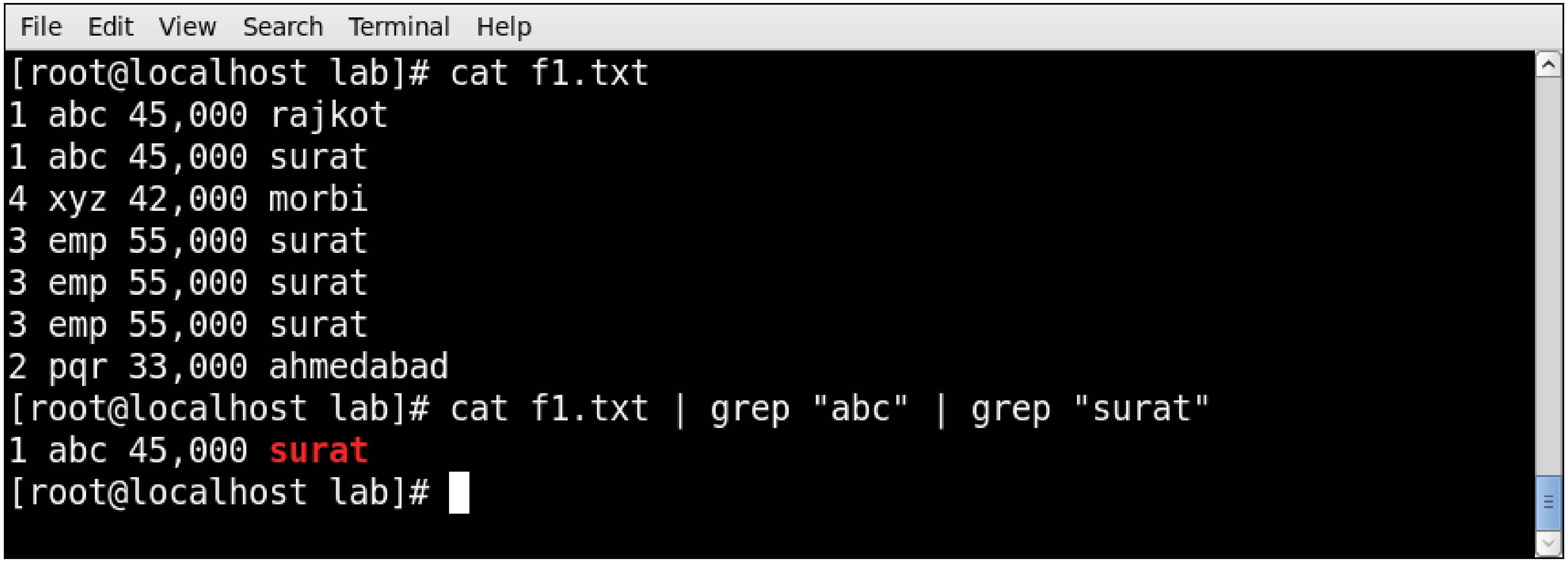
```
[root@localhost lab]# cat f1.txt
1 abc 45,000 rajkot
1 abc 45,000 rajkot
4 xyz 42,000 morbi
3 emp 55,000 surat
3 emp 55,000 surat
3 emp 55,000 surat
2 pqr 33,000 ahmedabad
[root@localhost lab]# uniq f1.txt
1 abc 45,000 rajkot
4 xyz 42,000 morbi
3 emp 55,000 surat
2 pqr 33,000 ahmedabad
[root@localhost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

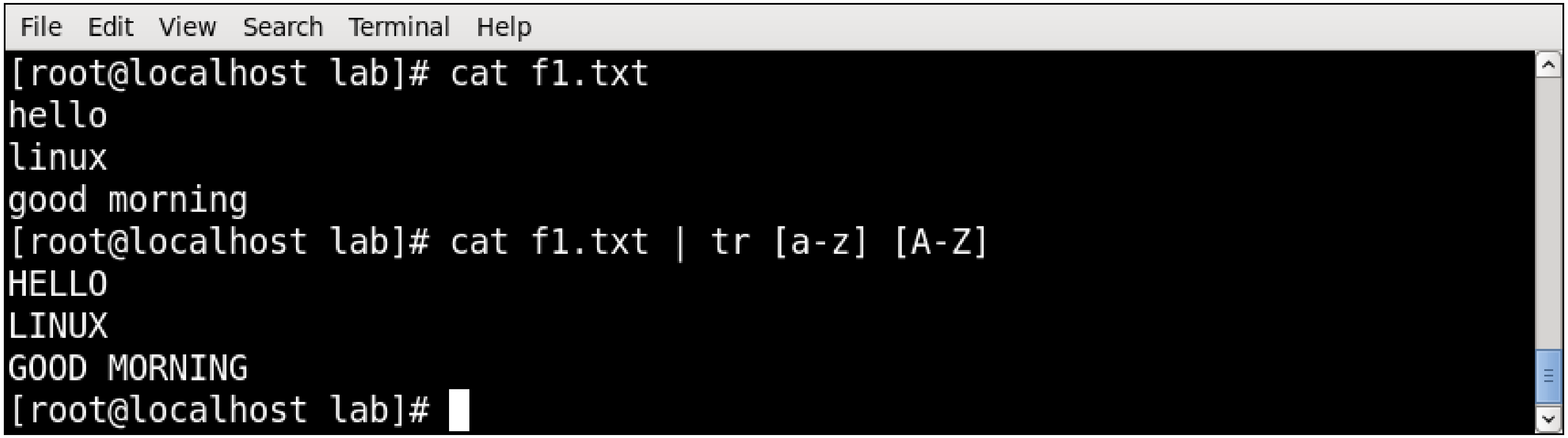
```
tr [OPTION] SET1 [SET2]
```

tr(translate) Command

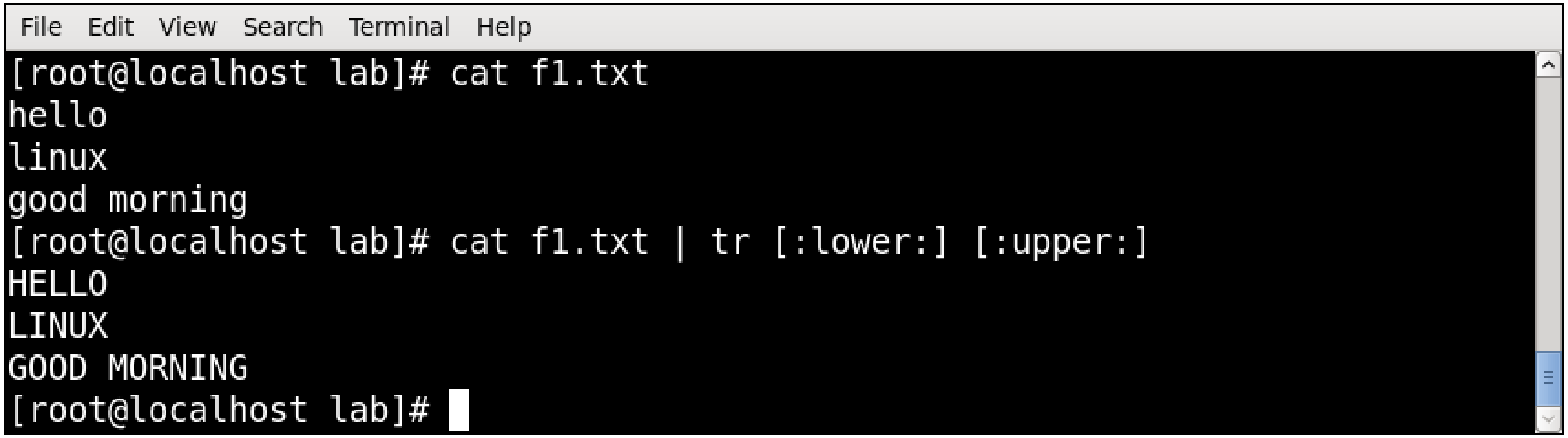
- **POSIX Character set supported by tr command :**
  - **[:digit:]** Only the digits 0 to 9.
  - **[:alnum:]** Any alphanumeric character.
  - **[:alpha:]** Any alpha character A to Z or a to z.
  - **[:blank:]** Space and TAB characters only.
  - **[:xdigit:]** Hexadecimal notation 0-9, A-F, a-f.
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

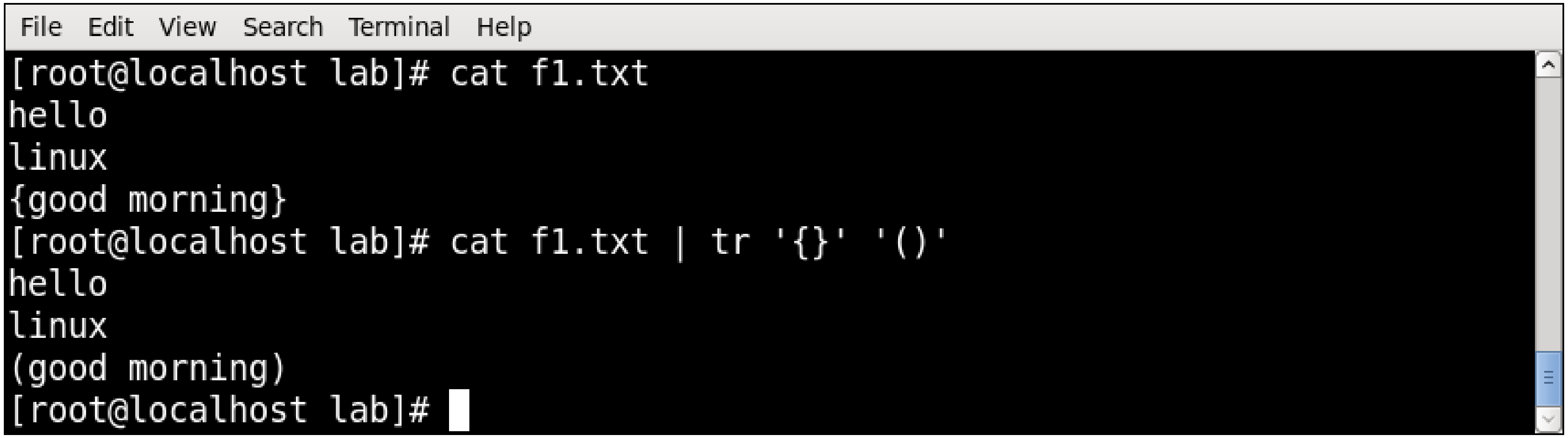


```
[root@localhost lab]# cat f1.txt
hello
linux
good morning
[root@localhost lab]# cat f1.txt | tr [a-z] [A-Z]
HELLO
LINUX
GOOD MORNING
[root@localhost lab]#
```



```
[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]#
```

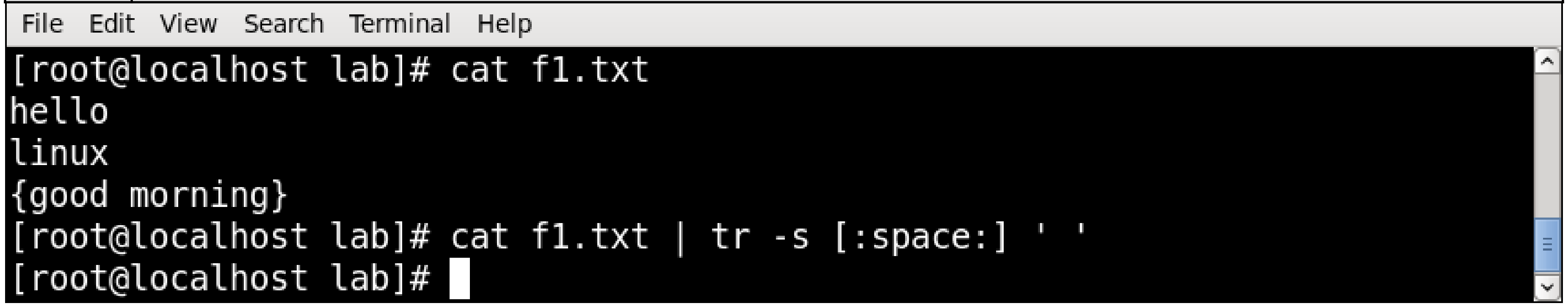
tr Command Example



```
[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

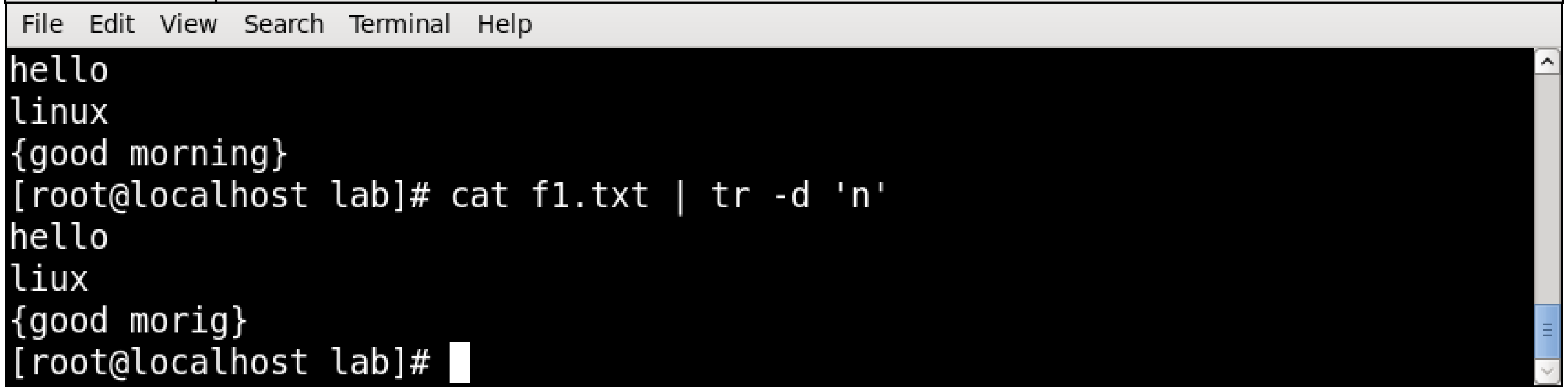
tr -s	**Replaces repeated characters listed in the set1 with single occurrence**



```
[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

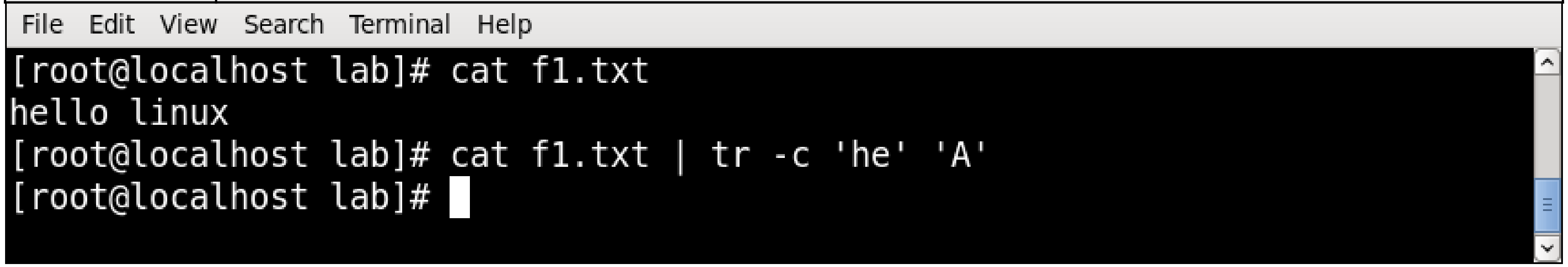
tr -d	**Delete characters in string1 from the input**



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

tr Command Example

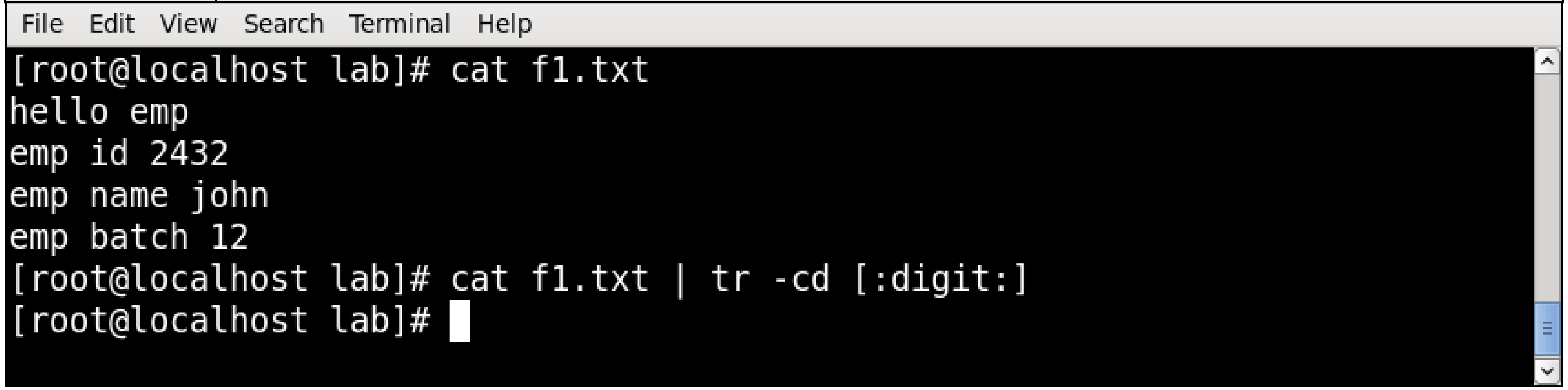
tr -c	**complements the set of characters in string.** **i.e., operations apply to characters not in the given set**



```
[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**

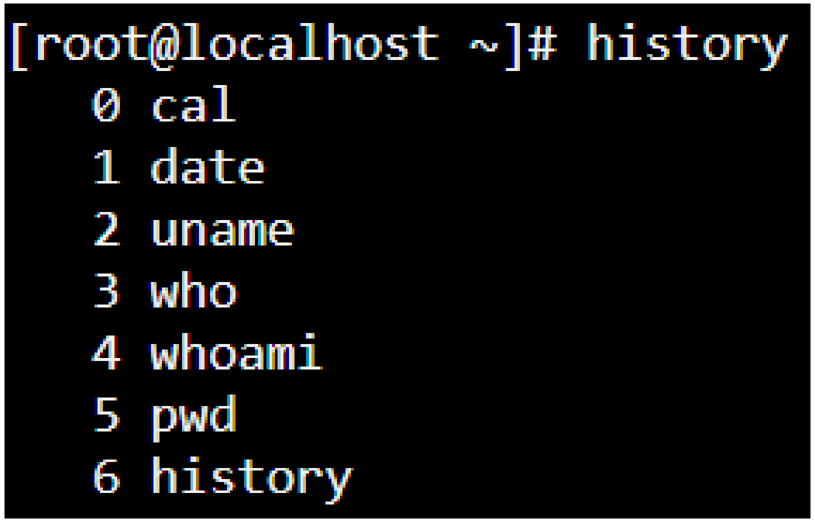


```
[root@localhost lab]# cat f1.txt
hello emp
emp id 2432
emp name john
emp batch 12
[root@localhost lab]# cat f1.txt | tr -cd [:digit:]
[root@localhost lab]#
```

history Command

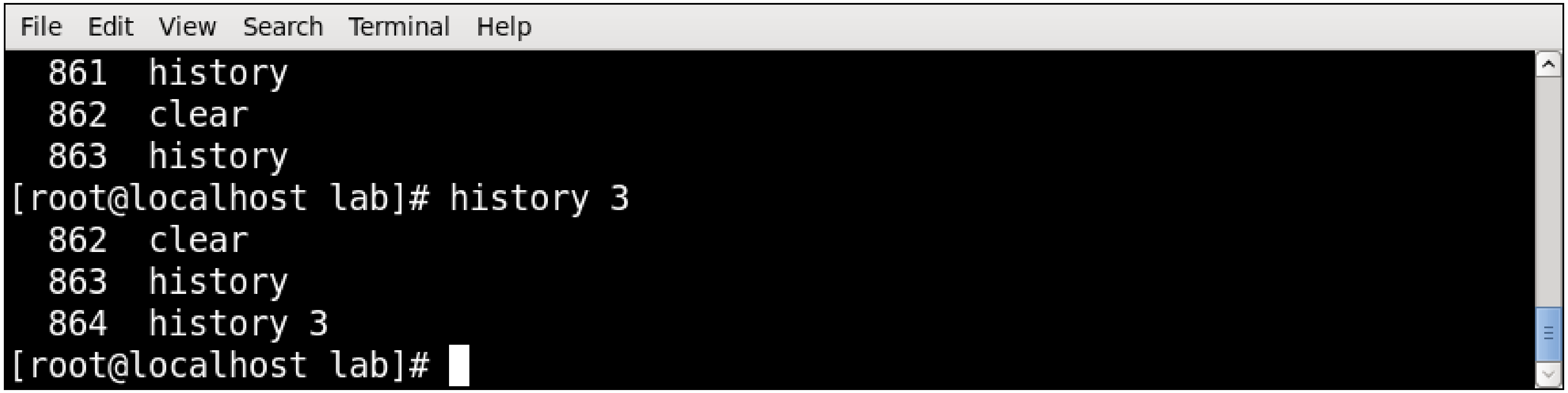
- **history** command is used to view the previously executed command.
- **Syntax :**

  history
- **Example :**

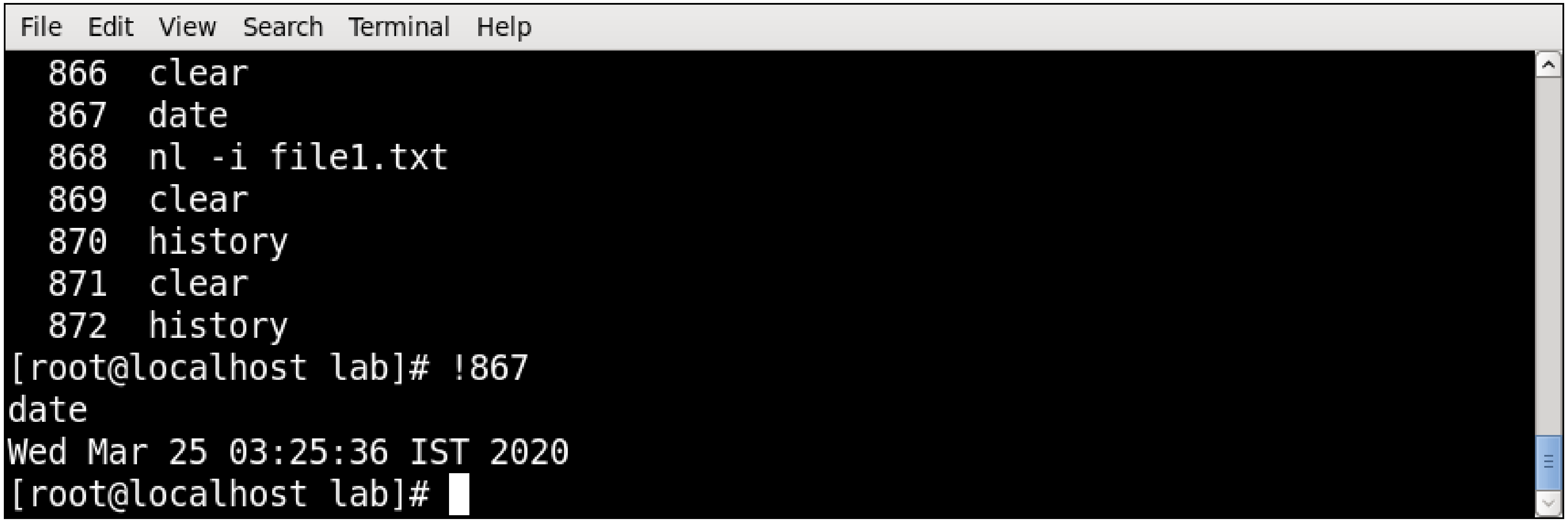


```
[root@localhost ~]# history
   0 cal
   1 date
   2 uname
   3 who
   4 whoami
   5 pwd
   6 history
```

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 :**

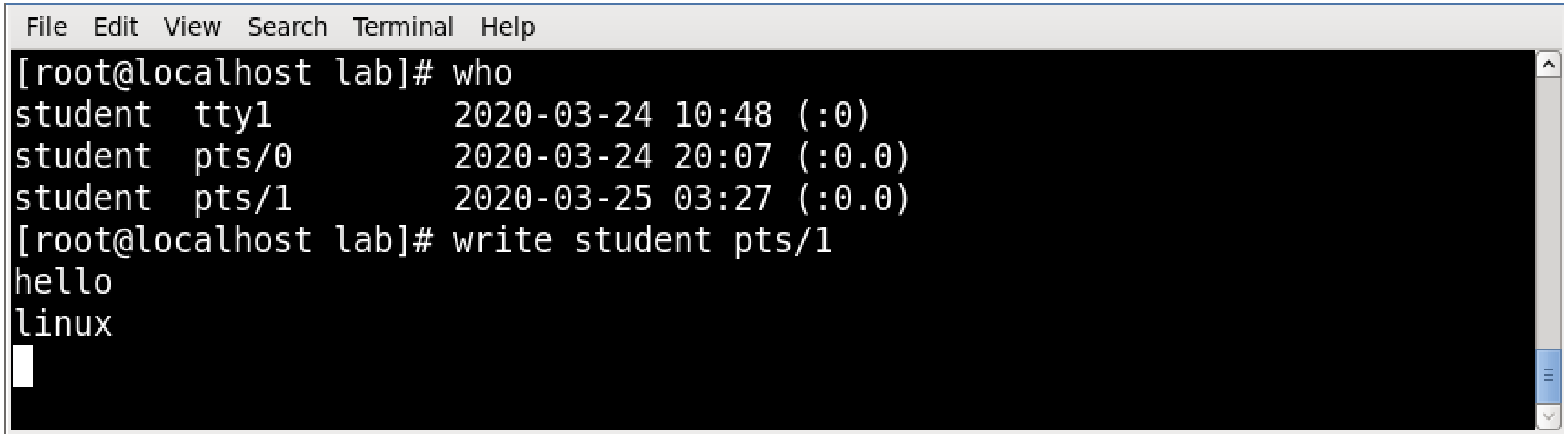
  write user [ttyname]

- **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

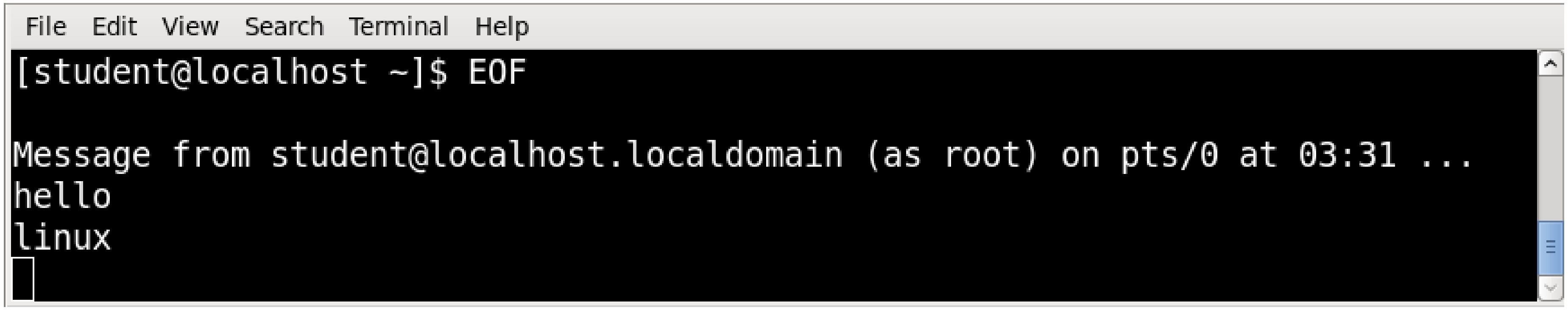
write Command Example

- 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 :**

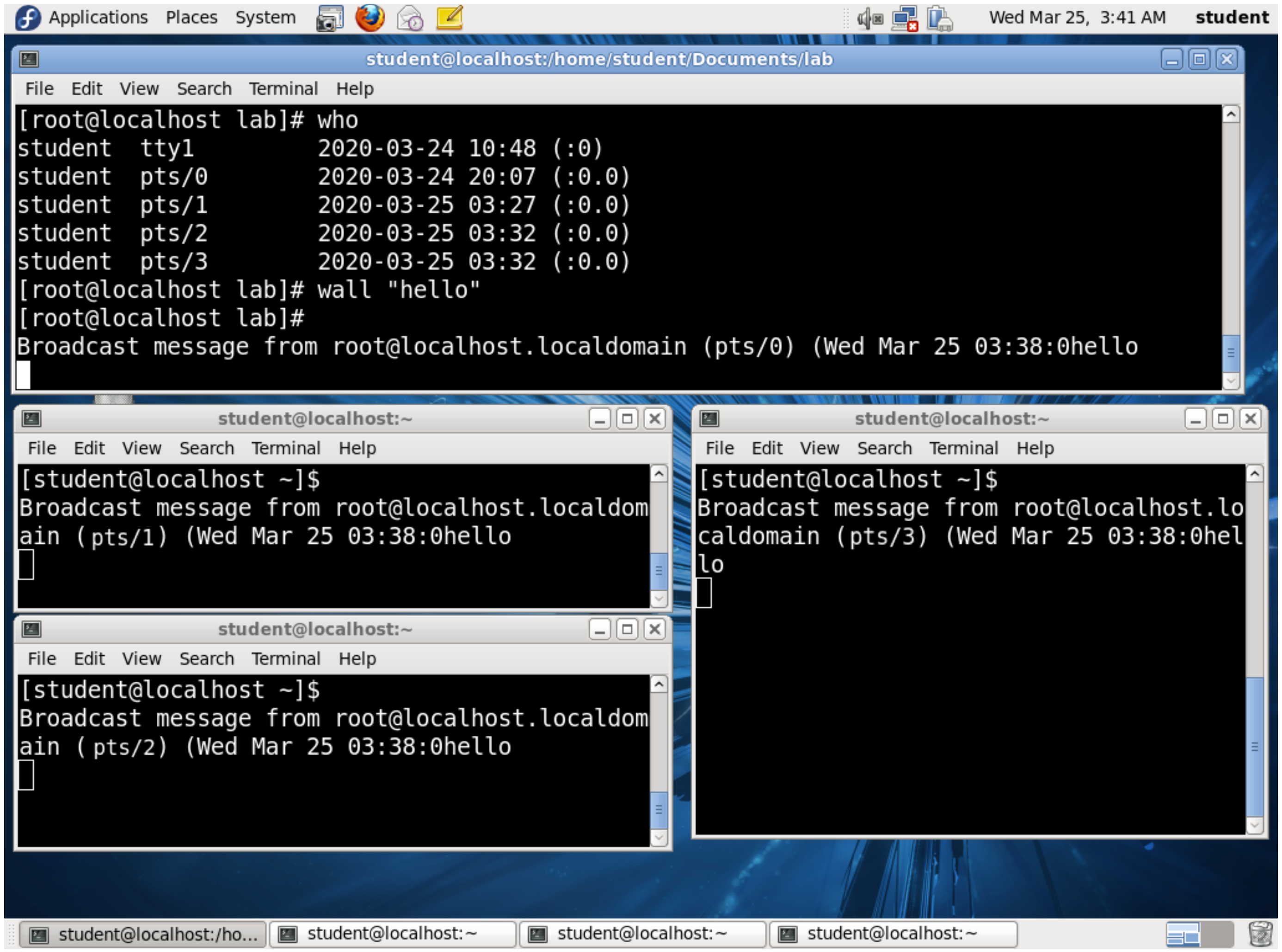
```
wall [-n] [-t TIMEOUT] [file]
```

- **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.


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

```
[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
```