LAB – 2
Operating Systems

GroupID - 6

Abhijeet Sonkar	IIB2019009
Aditya Aggarwal	IIT2019210
Ambika Singh	IIB2019017
Avneesh Kumar	IIB2019010
Divy Agrawal	IIT2019211

Questions Assigned

1F

Create your own SHELL-commands to simulate (mimic) the functionality of the well-known SHELL command cut .

2F

Write a SHELL Script which takes a source pathname/filename and a destination pathname from the user, checks whether the source file and the destination pathname (directory) exists. If not, prompts to the user to correctly provide the inputs. When it receives correct inputs, asks the user to input values for a delimiter and a field-number. It then executes your mycut command with the -d and -f options with the specified values of delimiter and field-number on the input file. The output of the execution is not displayed on the terminal; rather, it is redirected to a newly created file called output which is placed in the destination directory as mentioned by the user

Table of Contents

1

What is cut command?

Lets brielfy know about cut command.

The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by **byte position, character and field**. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is **not precedes** by its file name.

Syntax

```
$ cut OPTIONS… [FILE]…
```

The options that tell cut whether to use a delimiter, byte position, or character when cutting out selected portions the lines are as follows:

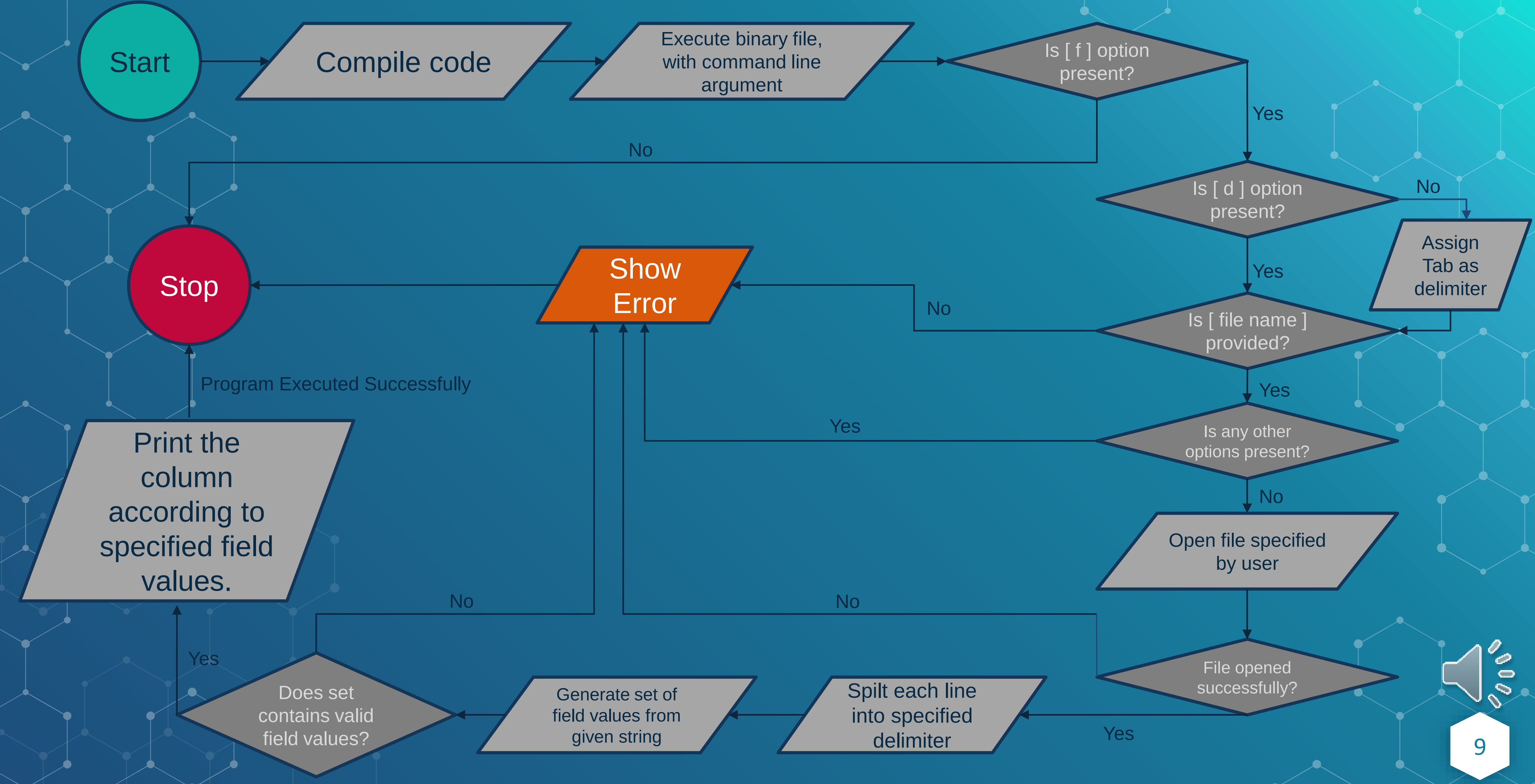
-f (--fields=LIST) - Select by specifying a field, a set of fields, or a range of fields. This is the most commonly used option.

-b (--bytes=LIST) - Select by specifying a byte, a set of bytes, or a range of bytes.

-c (--characters=LIST) - Select by specifying a character, a set of characters, or a range of characters.

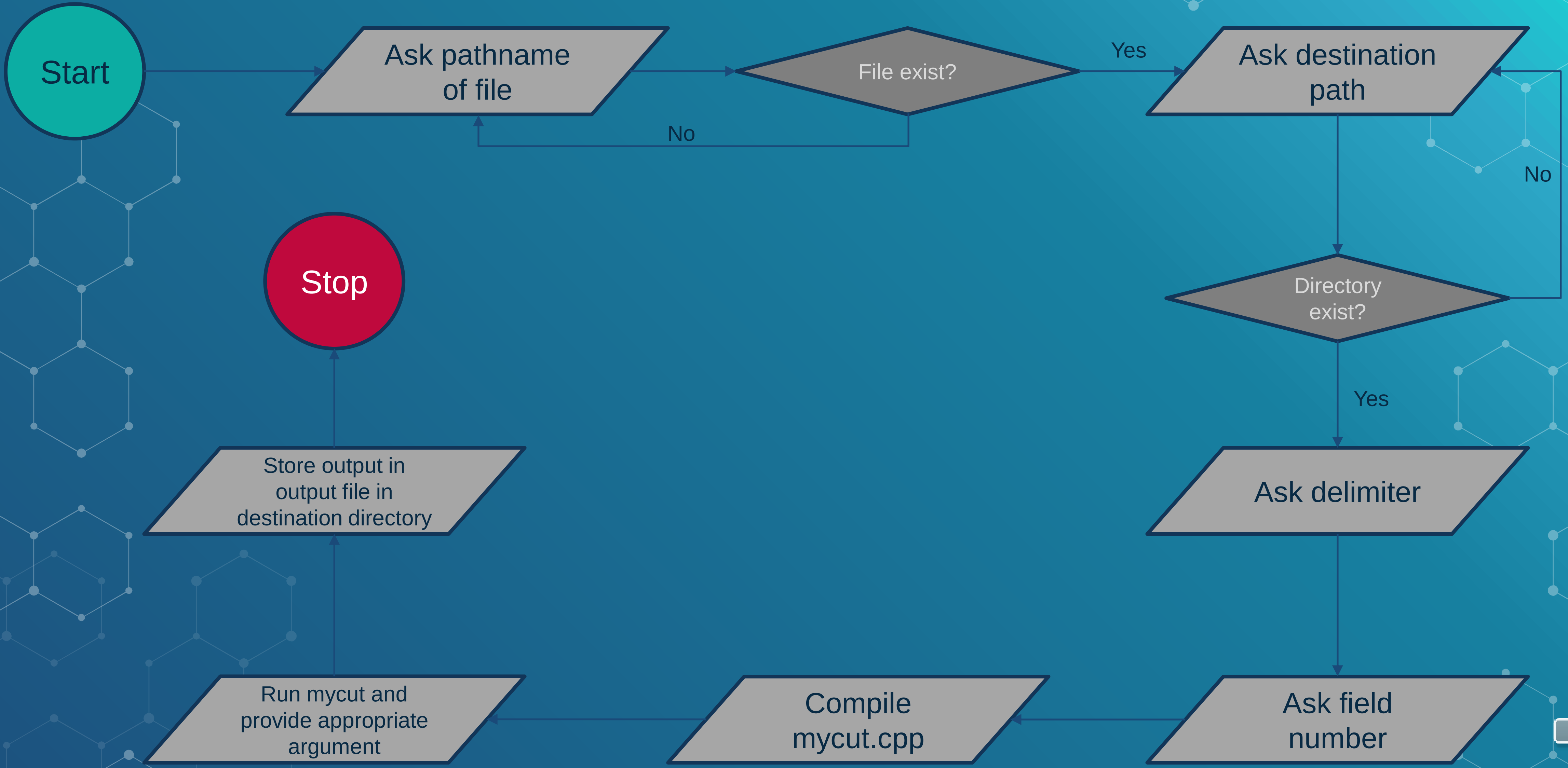
2 Flow Chart

mycut.cpp



Start
Compile code
Execute binary file, with command line argument
Is [f] option present?
Yes
No
Is [d] option present?
No
Assign Tab as delimiter
Yes
Stop
Show Error
No
Is [file name] provided?
Yes
Program Executed Successfully
Yes
Is any other options present?
No
Print the column according to specified field values.
Open file specified by user
No
No
Yes
File opened successfully?
Does set contains valid field values?
Generate set of field values from given string
Spilt each line into specified delimiter
Yes

script.bash


Start
Ask pathname of file
File exist?
No
Yes
Ask destination path
Directory exist?
No
Yes
Ask delimiter
Ask field number
Compile mycut.cpp
Run mycut and provide appropriate argument
Store output in output file in destination directory
Stop

3

Code Explanation

Lets brielfy discuss the code

```
int main(int argc, char **argv)
{

    bool DELEMITER_SPECIFIED = false;
    bool FIELD_NUMBER_SPECIFIED = false;
    int c;
    int digit_optind = 0;
    char delimiter;
    string str_fieldNumber;
```

In this portion of Program, we have declared some variable.

Here main() function is taking to arguments *argc* and *argv.*

```
            printf(": option(s) not available.\nTry using [d] and [f] options");
            exit(0);
        }
    }
```

- This part of code uses *getopt_long()* function to fetch options and their argument values by parsing the array of arguments i.e. *argv*.
- An element of *argv* that starts with '-' is an option element.
- When *getopt_long()* is called repeatedly, it returns successively each of the option characters from each of the option elements.
- And it returns -1 when there are no more option characters left in *argv*.
- *“d:f:”* here is termed as *optstring*, which shows that d and f are options and they both require an argument.

```
if (FIELD_NUMBER_SPECIFIED)
{
    char buffer;
    int i = 0;
    string line;
    char delim = (DELEMITER_SPECIFIED == true) ? delimiter : '\t';
    vector<string> splitWords;
    int fileIndex = (DELEMITER_SPECIFIED) ? 5 : ((!DELEMITER_SPECIFIED) ? 3 : argc - 1);
    if (argv[fileIndex] == NULL)
    {
        cout << "mycut: file pathname not provided." << endl;
        exit(0);
    }
    int fd = open(argv[fileIndex], O_RDONLY);

    if (fd == -1) // file DNE
    {
        perror(argv[fileIndex]);
    }
```

- If delimiter is specified by the user then store it in ***delim*** else store ***tab*** as it's default value.
- If the delimiter is provided by the user then filename is stored at ***5th*** position else stored at ***3rd*** position in the argument list.
- If filename is not specified or is not valid then print error and exit.
- If filename is valid open it using ***syscall***.

```
                    else
                    {
                        cout << printWord[i] << delim;
                    }
                }
                cout << endl;
                line.clear();
                splitWords.clear();
                printWord.clear();
            }
        }
        else
        {
            line.push_back(buffer);
        }
        if (endOfFile == 0)
        {
            break;
        }
    }
}
```

- Syscall to read file character by character and store it in a buffer.
- When the line does not contain the delimiter then print the whole line.
- Split the line according to the delimiter and store the word in spliwords Array.
- All the word to be printed are stored in printWord array
- Printing the whole array printword array
- Breaking the loop when end of the file is reached.

Explanation of generateFieldNumbers() function

```
generateFieldNumber(string fieldString,int maxLength)
```

This function basically checks whether the given fieldvalue is valid or not. If valid, it then process the string and extract range of numbers from the string. eg. Of some invalid fieldvalues are alphabets, symbol other than comma, hyphen, 0 as one of its value.
4 types of values can be provided in field string, and we process each type of fieldvalues individually and insert the values in a set.

```
set<int> fieldNumbers;
```

The 4 type of field values are :-

- *Normal integer values:* eg. 3,4,45..

```
        if (count(i.begin(), i.end(), '-') == 0)
            {
                int temp = atoi(i.c_str());
                fieldNumbers.insert(temp);
            }
```

- ***Range value which start with hyphen at the start***

  Like -5, -7 ,- 10 etc

  -5 basically means 1,2,3,4,5

```
if (*i.begin() == '-') {
                i.erase(i.begin());
                int temp = atoi(i.c_str());
                for (int i = 1; i <= temp; i++) {
                    fieldNumbers.insert(i);
                }
            }
```

- ***Range with end with hyphen like***

  4-,5- ...etc

  4- mean 4,5,6,7,...... till the end of the column

  - Here maxLength is the length of the column.

```
     i.pop_back();
     int temp = atoi(i.c_str());
     minField = min(minField, temp)
 for (int i = minField; i <= maxLength; i++){
      fieldNumbers.insert(i);
 }
```

- ***Range with hyphen at the middle***

  Like 4-7,3-10…...etc; 4-7 means 4,5,6,7

```
string firstString;
               int a = 0;
               while (i[a] != '-')
               {
                   firstString.push_back(i[a]);
                   a++;
               }
               int firstNumber = atoi(firstString.c_str());
               string secondString;
               while (a < i.size())
               {
                   secondString.push_back(i[a]);
                   a++;
               }
               secondString.erase(secondString.begin());

               int secondNumber = atoi(secondString.c_str());

               if (secondNumber < firstNumber)
               {
                   validField = false;
               }
               for (int x = firstNumber; x <= secondNumber; x++)
               {
                   fieldNumbers.insert(x);
               }
```

Bash code Explanation

```
#!/bin/bash
IFS= read -r -p "Please Enter the absolute pathname of the file: " pathname
unset IFS
if [ ! -f "$pathname" ];
then
    echo "No such file exists!!!!"
    flag=0
    while [ $flag -eq 0 ]
    do
        IFS= read -r -p "Please Enter the correct absolute pathname of the file: " pathname
        unset IFS
        if [ -f "$pathname" ];
        then
            flag=`expr $flag+1`
            echo "file found!"
            break
        else
            echo "No such file exists!!!!"
            continue
        fi
    done
fi
```

-r option of read command is used to take input without interpreting the backslash escapes like '\n', '\t', '\b' etc. So basically it is used here as the pathname of file or directory may contain backslash characters. And IFS option before read command is to prevent leading/trailing whitespace from being trimmed. -p option output the string prompt message without a trailing newline before attempting to read the input to be given by the user. so `IFS= read -r -p msg var_name` *is used here to display the prompt message and read the absolute pathname of file on which we have apply cut command until a newline character or enter key is pressed by the user. then we will unset the environment variable IFS.*

Now we will check whether the absolute pathname of the file given by the user is correct or not using the -f option which returns true value if file exists and false if not. if the pathname entered by the user is correct then rest of the code here is skipped. but if the pathname entered by the user is incorrect then we will ask for the correct pathname until we receive the correct pathname using the while loop. here the value 0 of the variable named flag denotes that we have not received correct pathname till now and value 1 denotes that correct pathname has been received. And while loop will run until correct pathname is received. And as soon as correct pathname is received the flag variable becomes 1, "file found " is printed and we break the while loop.

```
IFS= read -r -p "Please Enter the absolute destination pathname: " destination_name
unset IFS
if [ ! -d "$destination_name" ];
then
    echo "No such directory exits!!!!"
    flag=0
    while [ $flag -eq 0 ]
    do
        IFS= read -r -p "Please Enter the correct absolute destination name: " destination_name
        unset IFS
        if [ -d "$destination_name" ];
        then
            flag=`expr $flag+1`
            echo "directory found!"
            break
        else
            echo "No such directory exits!!!!"
            continue
        fi
    done
fi
```

similarly as explained in previous slide, `IFS= read -r -p msg var_name` *is used here to display the prompt message to enter the absolute destination pathname where output file is to be stored. and read the absolute pathname of destination directory until a newline character or enter key is pressed by the user. this pathname is stored in the variable named destination name. then we will unset the environment variable IFS.*

similarly as explained in previous slide, here also we will check whether the destination pathname inputted by the user is correct or not using the -d option which returns true if the directory exists and false if doesn't exist. if the inputted destination pathname is correct then the rest of code here also is skipped. But if the inputted destination pathname is incorrect then we will ask user again and again to enter the correct destination pathname until he enters the correct path using the while loop. similarly here the variable named flag has the same purpose as before and as soon as correct destination path is entered flag variable becomes 1, "directory found " is printed and we break the while loop.

```
IFS= read -r -p "Please enter the Delimiter: " delimiter
unset IFS
read -p "Please enter the fieldnumber: " fieldnumber
out="/output"
destination_file=$destination_name$out
`touch "$destination_file"`
`g++ mycut.cpp -o mycut`
`./mycut - d "$delimiter" -f $fieldnumber "$pathname" > "$destination_file"`
exit
```

Now we will ask the user to enter the desired delimiter normally without double quotes. here also we will use -r option to ignore the interpretation of the backslash escape characters and take input until enter is pressed and -p option to display the prompt message. delimiter is basically a sequence of one or more characters for specifying the boundary between separate, independent regions in plain text. the entered delimiter is stored in the variable named delimiter. then we will unset the environment variable IFS.

Now we will ask the user to enter the field number and store in the variable named fieldnumber. the -p option is used to prompt a message to enter the field number.

the variable named destination_file contains the absolute pathname of the output file which will contain the output of the simulated cut command. this pathname is obtained by concatenating the absolute pathname for destination directory and the name of the file that is output. this concatenation is done above as $destination_name$out where variable named out contains the name of file that is output. then we will create the file named output using the touch command.

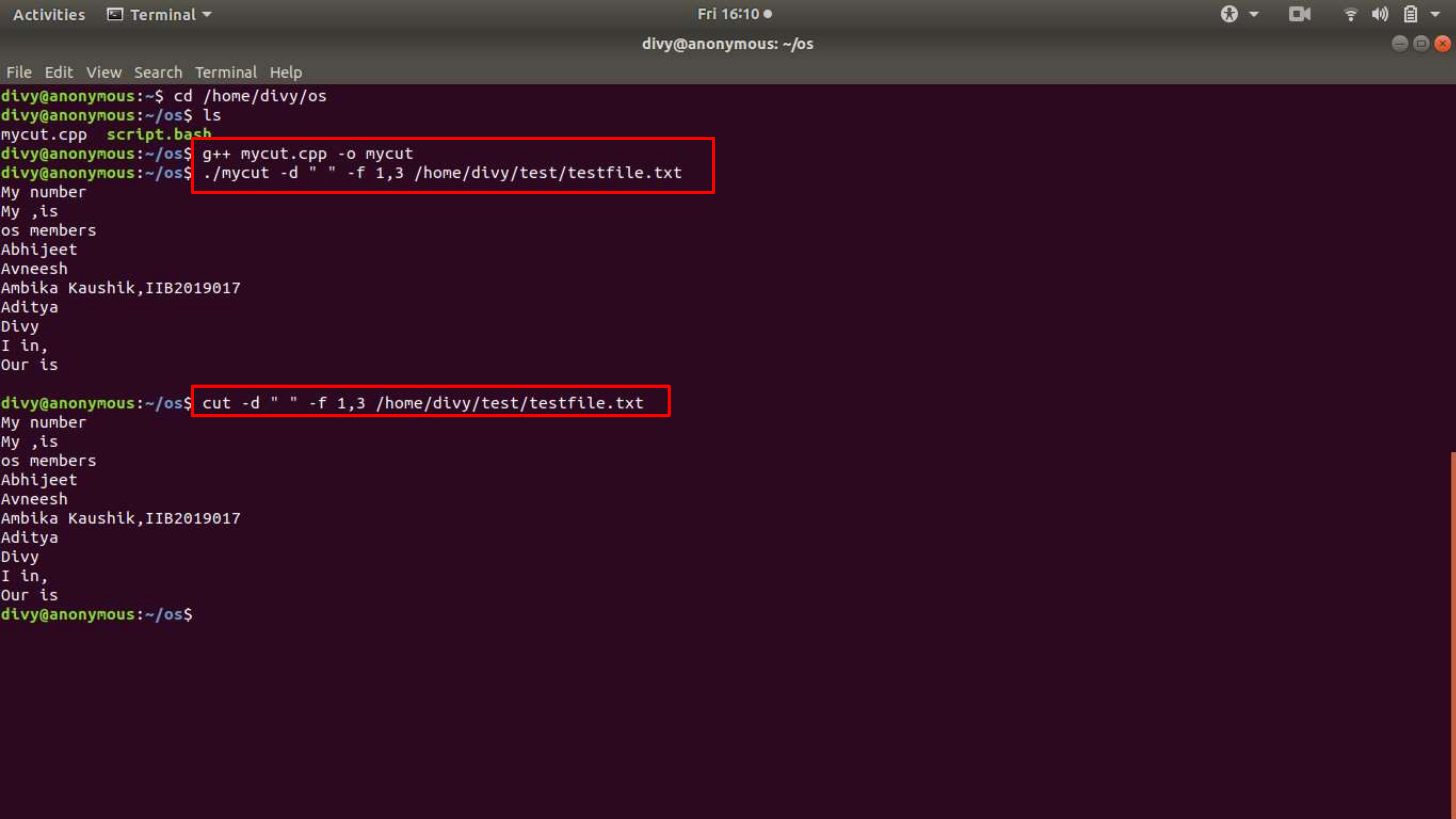
now we will compile the code in mycut.cpp which mimics the linux command cut using g++ and name the executable output file as mycut using the -o option. then we will run the executable output file created.

4

Output Screenshots

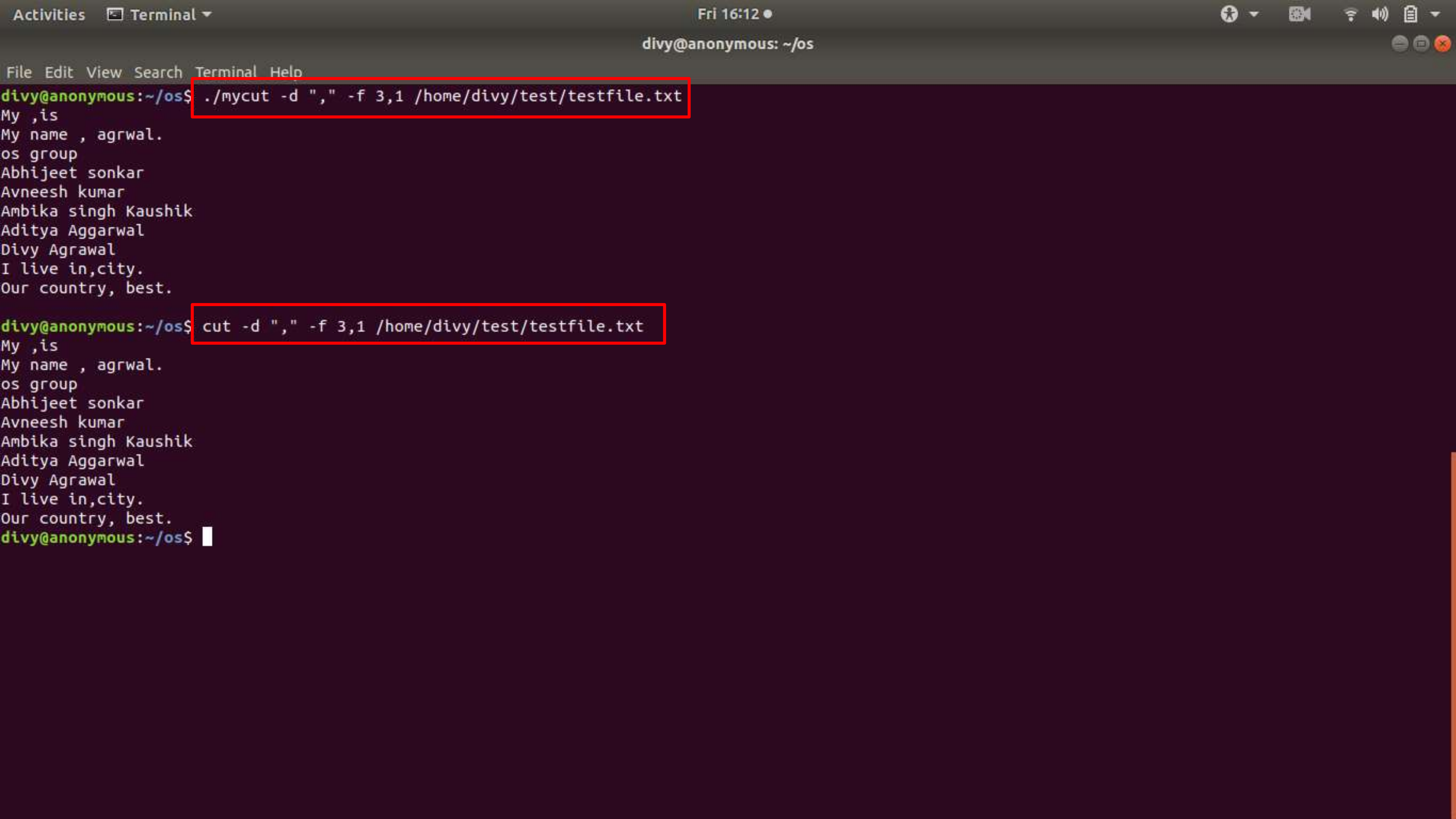
Lets have a look at the screenshots

Output of mycut.cpp



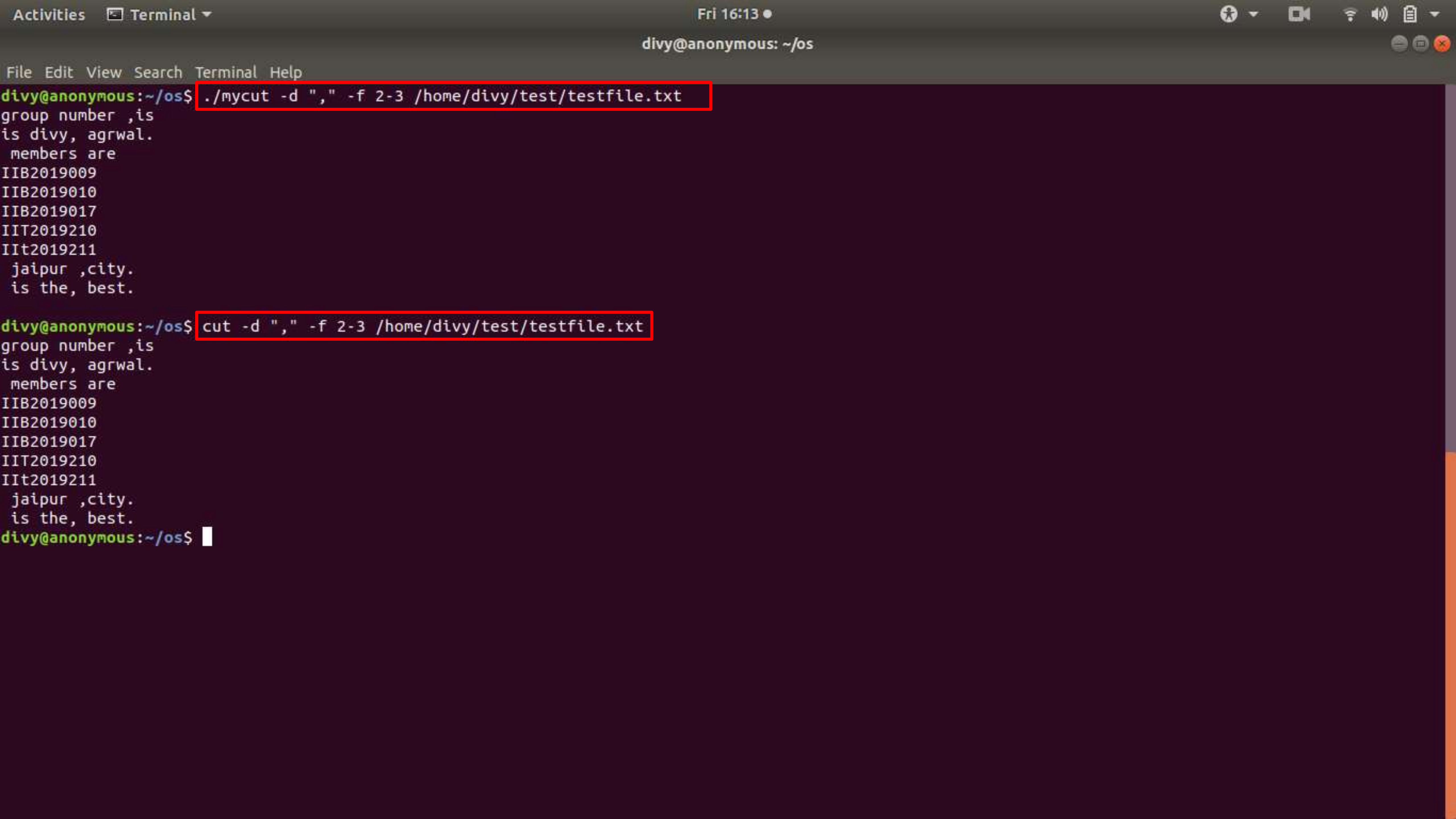
```
divy@anonymous:~$ cd /home/divy/os
divy@anonymous:~/os$ ls
mycut.cpp  script.bash
divy@anonymous:~/os$ g++ mycut.cpp -o mycut
divy@anonymous:~/os$ ./mycut -d " " -f 1,3 /home/divy/test/testfile.txt
My number
My ,is
os members
Abhijeet
Avneesh
Ambika Kaushik,IIB2019017
Aditya
Divy
I in,
Our is

divy@anonymous:~/os$ cut -d " " -f 1,3 /home/divy/test/testfile.txt
My number
My ,is
os members
Abhijeet
Avneesh
Ambika Kaushik,IIB2019017
Aditya
Divy
I in,
Our is
divy@anonymous:~/os$
```



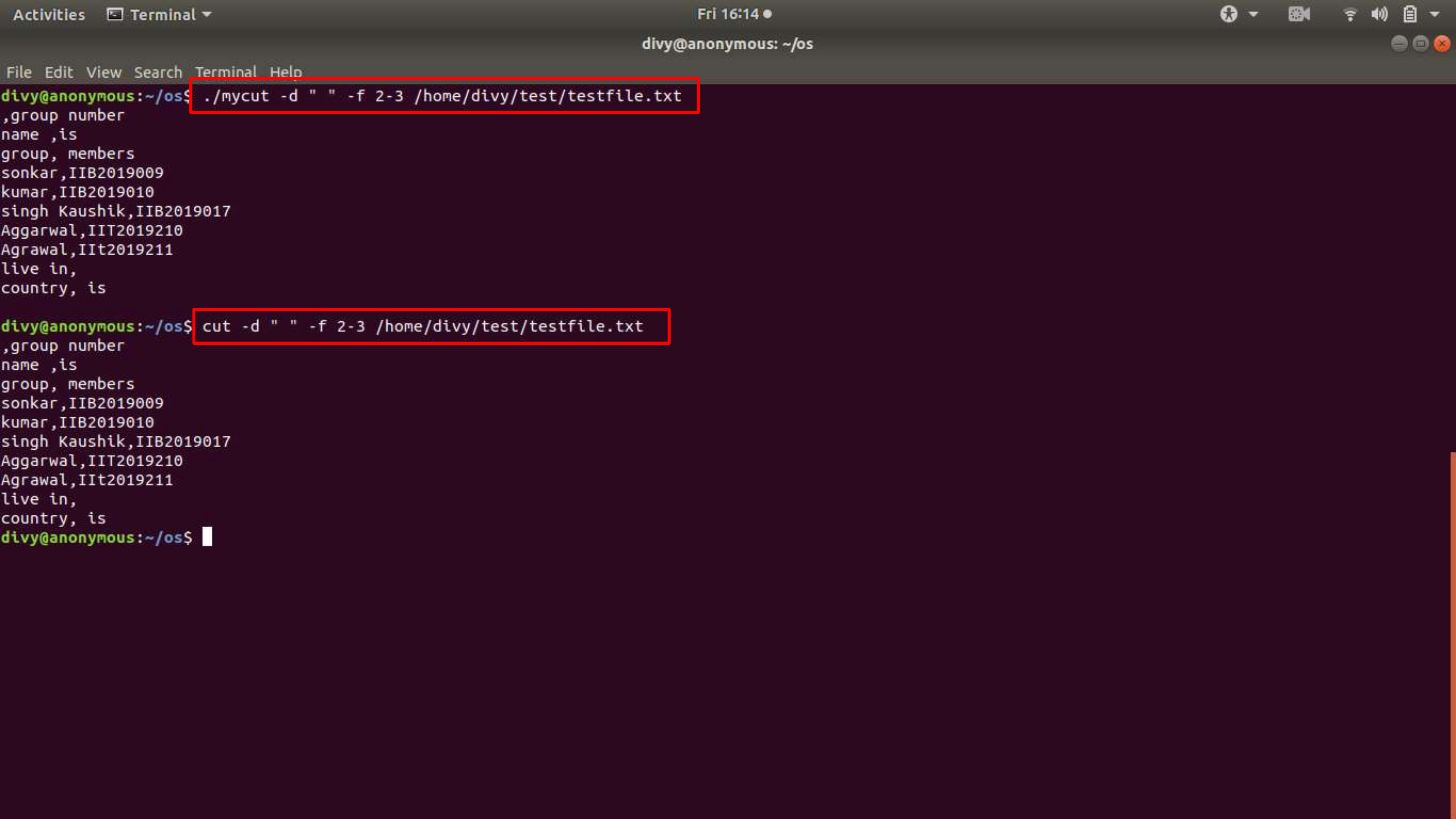
```
divy@anonymous:~/os$ ./mycut -d "," -f 3,1 /home/divy/test/testfile.txt
My ,is
My name , agrwal.
os group
Abhijeet sonkar
Avneesh kumar
Ambika singh Kaushik
Aditya Aggarwal
Divy Agrawal
I live in,city.
Our country, best.

divy@anonymous:~/os$ cut -d "," -f 3,1 /home/divy/test/testfile.txt
My ,is
My name , agrwal.
os group
Abhijeet sonkar
Avneesh kumar
Ambika singh Kaushik
Aditya Aggarwal
Divy Agrawal
I live in,city.
Our country, best.
divy@anonymous:~/os$ 
```



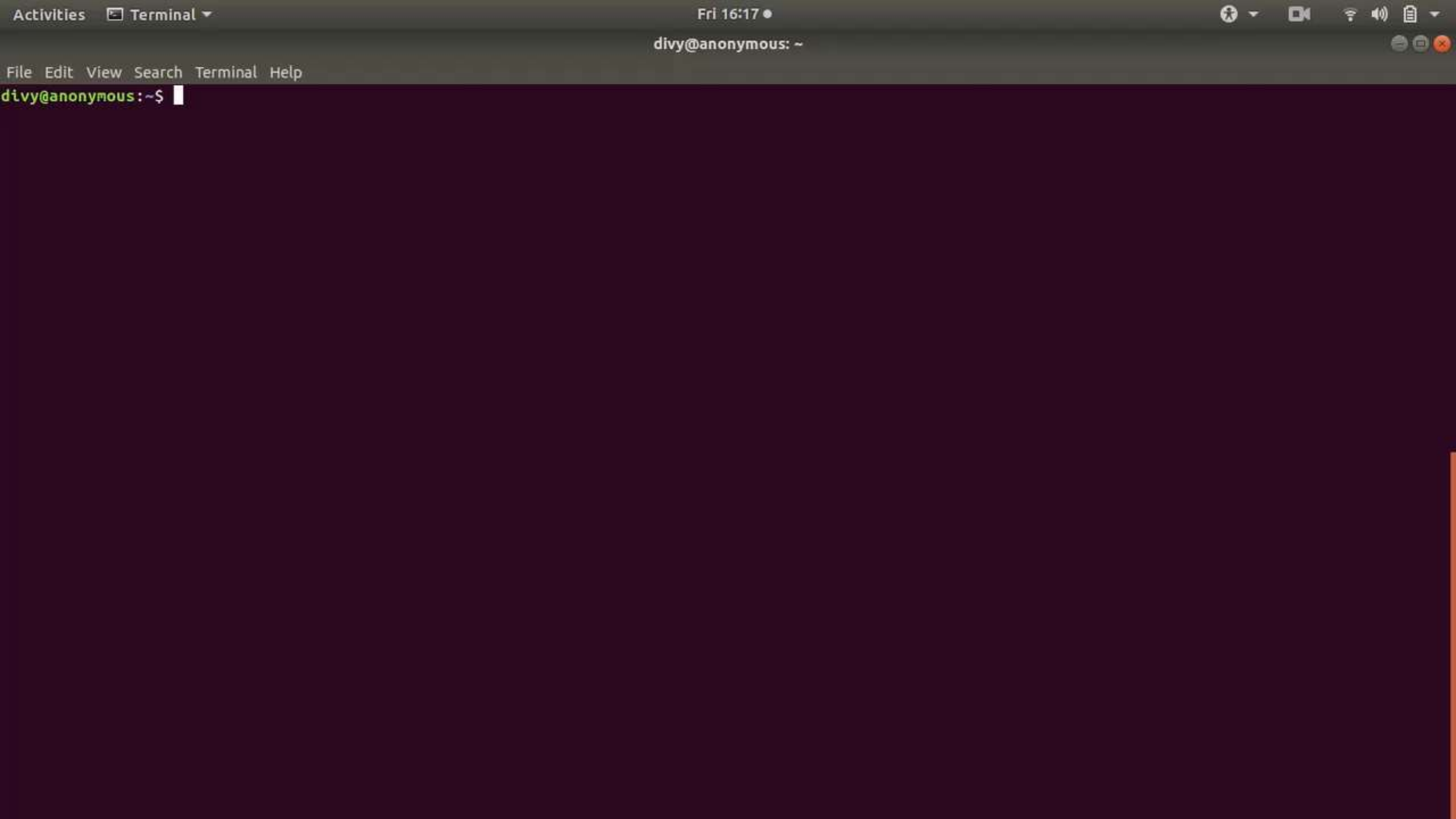
```
divy@anonymous:~/os$ ./mycut -d "," -f 2-3 /home/divy/test/testfile.txt
group number ,is
is divy, agrwal.
 members are
IIB2019009
IIB2019010
IIB2019017
IIT2019210
IIt2019211
 jaipur ,city.
 is the, best.

divy@anonymous:~/os$ cut -d "," -f 2-3 /home/divy/test/testfile.txt
group number ,is
is divy, agrwal.
 members are
IIB2019009
IIB2019010
IIB2019017
IIT2019210
IIt2019211
 jaipur ,city.
 is the, best.
divy@anonymous:~/os$ 
```



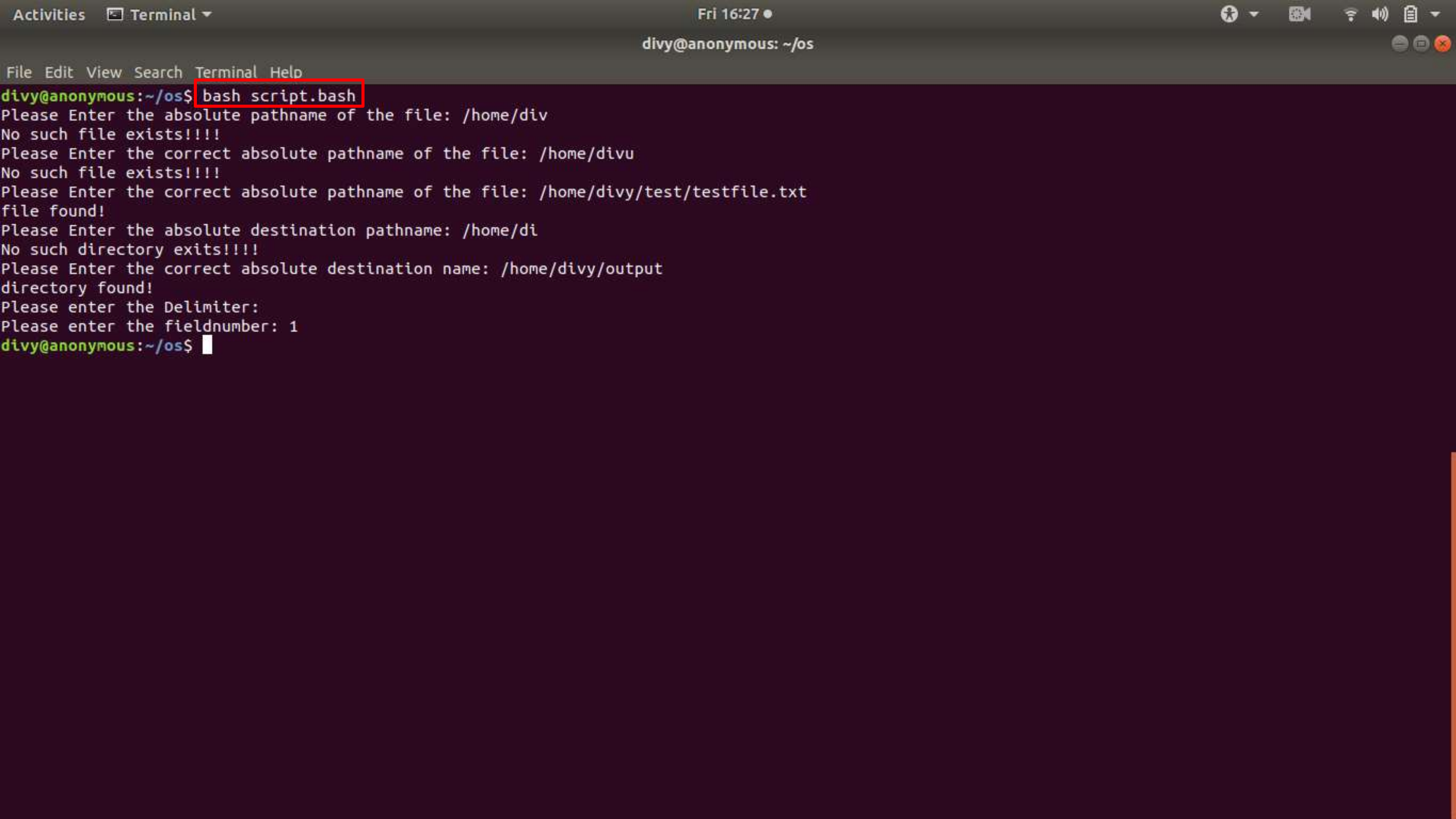
```
divy@anonymous:~/os$ ./mycut -d " " -f 2-3 /home/divy/test/testfile.txt
,group number
name ,is
group, members
sonkar,IIB2019009
kumar,IIB2019010
singh Kaushik,IIB2019017
Aggarwal,IIT2019210
Agrawal,IIt2019211
live in,
country, is

divy@anonymous:~/os$ cut -d " " -f 2-3 /home/divy/test/testfile.txt
,group number
name ,is
group, members
sonkar,IIB2019009
kumar,IIB2019010
singh Kaushik,IIB2019017
Aggarwal,IIT2019210
Agrawal,IIt2019211
live in,
country, is
divy@anonymous:~/os$ 
```

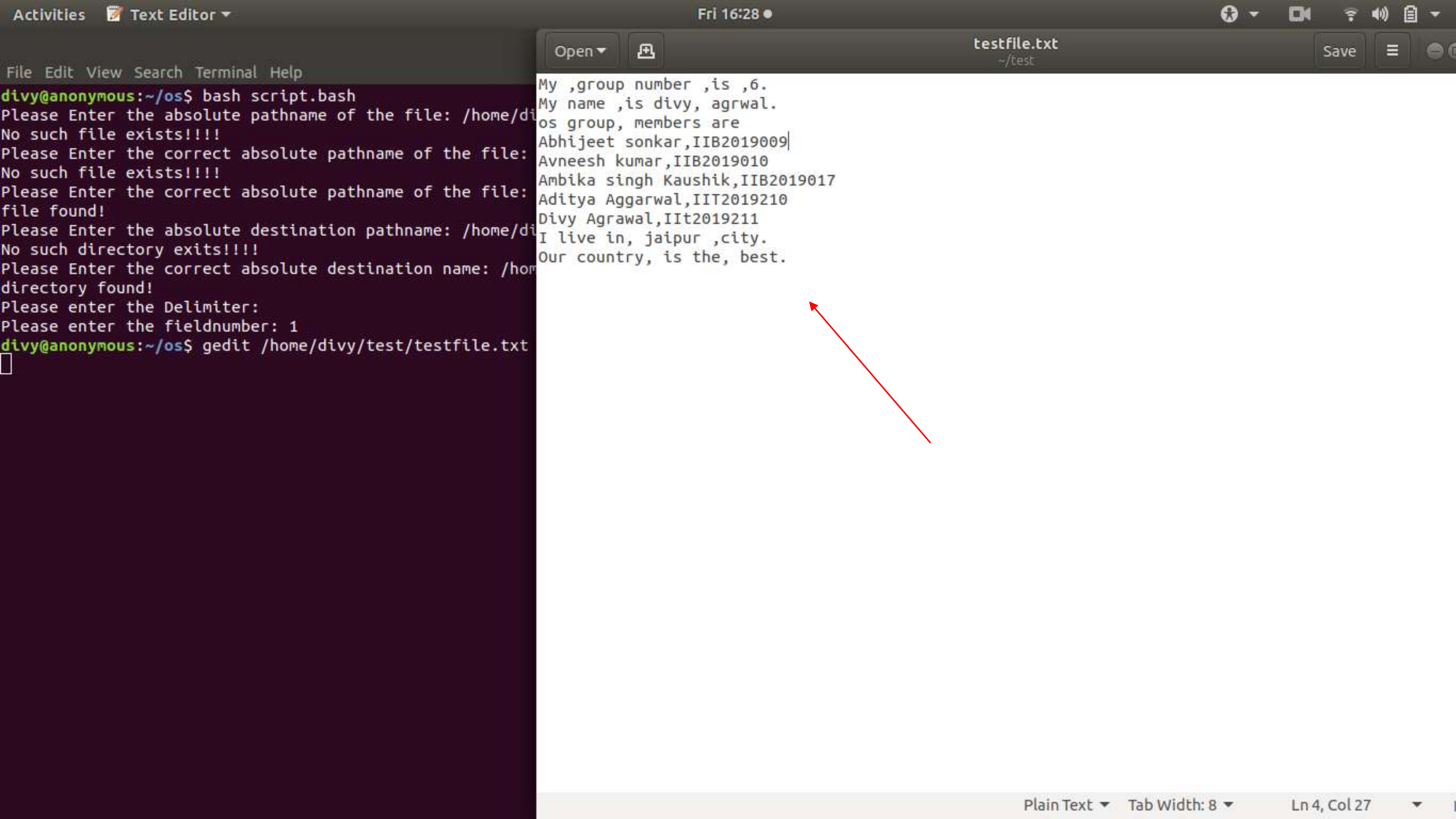


```
divy@anonymous:~$ 
```

Output of script.bash

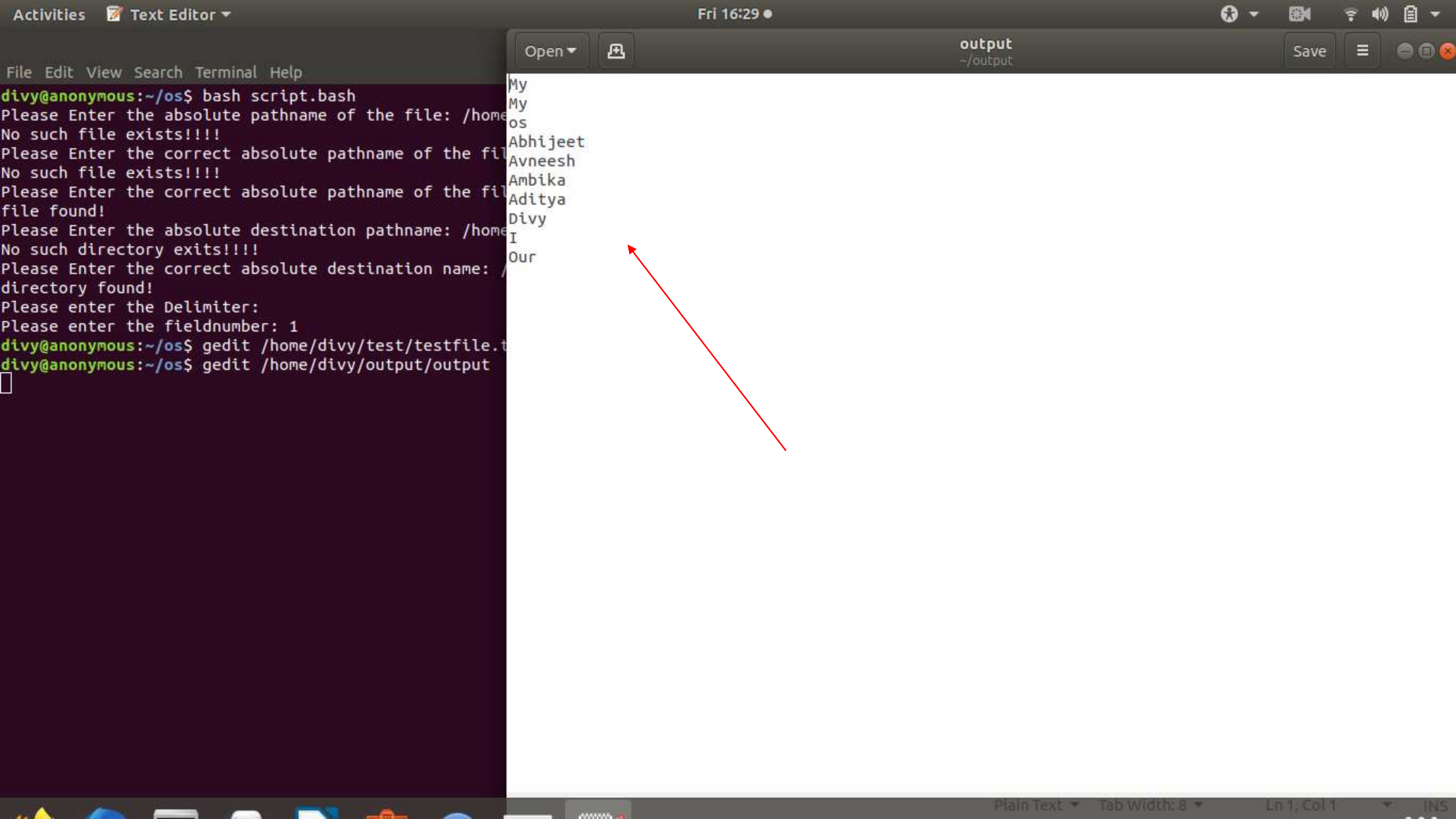


```
divy@anonymous:~/os$ bash script.bash
Please Enter the absolute pathname of the file: /home/div
No such file exists!!!!
Please Enter the correct absolute pathname of the file: /home/divu
No such file exists!!!!
Please Enter the correct absolute pathname of the file: /home/divy/test/testfile.txt
file found!
Please Enter the absolute destination pathname: /home/di
No such directory exits!!!!
Please Enter the correct absolute destination name: /home/divy/output
directory found!
Please enter the Delimiter:
Please enter the fieldnumber: 1
divy@anonymous:~/os$
```



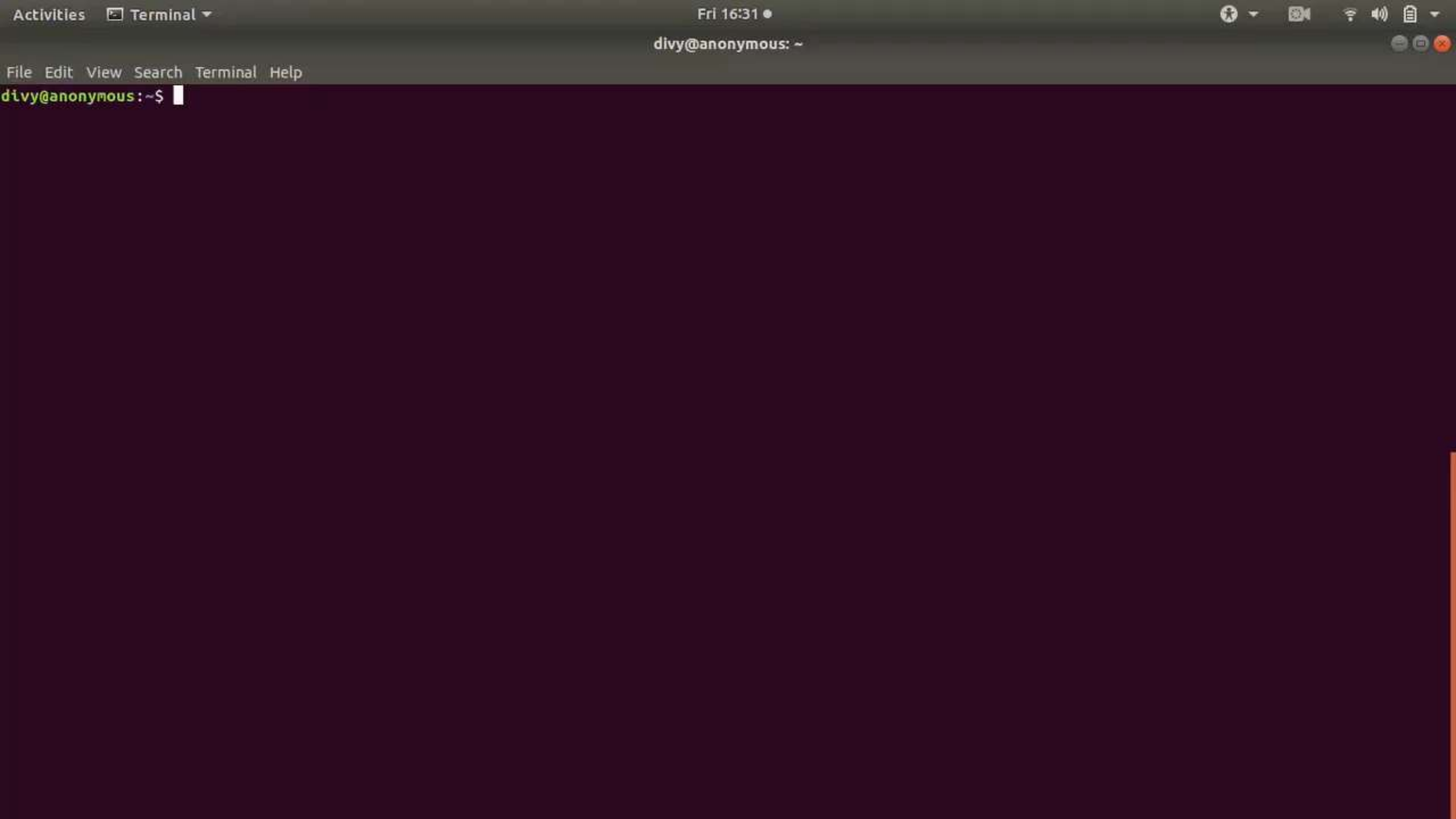
```
divy@anonymous:~/os$ bash script.bash
Please Enter the absolute pathname of the file: /home/di
No such file exists!!!!
Please Enter the correct absolute pathname of the file:
No such file exists!!!!
Please Enter the correct absolute pathname of the file:
file found!
Please Enter the absolute destination pathname: /home/di
No such directory exits!!!!
Please Enter the correct absolute destination name: /hom
directory found!
Please enter the Delimiter:
Please enter the fieldnumber: 1
divy@anonymous:~/os$ gedit /home/divy/test/testfile.txt
```

```
My ,group number ,is ,6.
My name ,is divy, agrwal.
os group, members are
Abhijeet sonkar,IIB2019009
Avneesh kumar,IIB2019010
Ambika singh Kaushik,IIB2019017
Aditya Aggarwal,IIT2019210
Divy Agrawal,IIt2019211
I live in, jaipur ,city.
Our country, is the, best.
```



```
divy@anonymous:~/os$ bash script.bash
Please Enter the absolute pathname of the file: /home
No such file exists!!!!
Please Enter the correct absolute pathname of the fil
No such file exists!!!!
Please Enter the correct absolute pathname of the fil
file found!
Please Enter the absolute destination pathname: /home
No such directory exits!!!!
Please Enter the correct absolute destination name: /
directory found!
Please enter the Delimiter:
Please enter the fieldnumber: 1
divy@anonymous:~/os$ gedit /home/divy/test/testfile.t
divy@anonymous:~/os$ gedit /home/divy/output/output
```

```
My
My
os
Abhijeet
Avneesh
Ambika
Aditya
Divy
I
Our
```



```
divy@anonymous:~$ 
```

THANK YOU