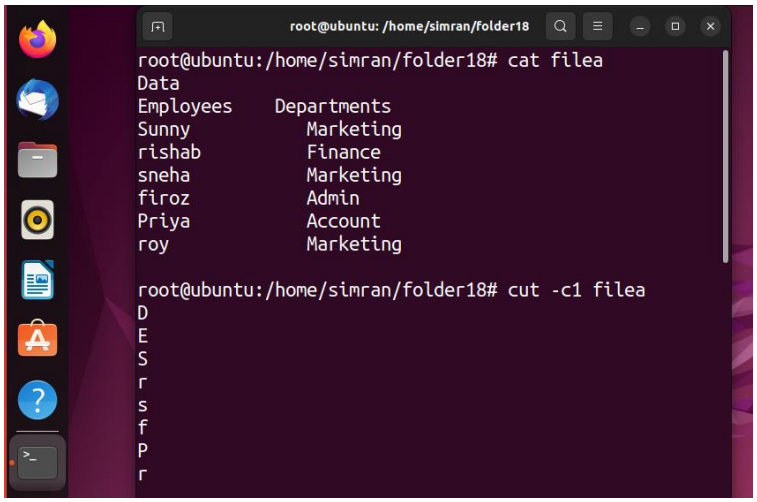
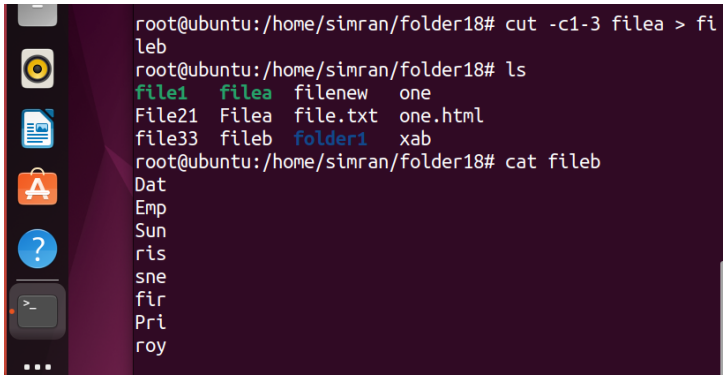
Filters / Text Processing Commands

Filters and text processing commands are often used to manipulate and process text data efficiently

CUT Commands:



Now storing these first 3 characters from filea to a new file fileb



AWK Command:

The `awk` command is a powerful text processing tool and programming language that is commonly used in Unix and Unix-like operating systems. It operates on a per-line basis, processing text line by line, making it well-suited for handling structured data. `awk` allows you to define patterns and actions to be taken when those patterns are matched.

AWK command considers each line in file as a row and each word as a field/column

Here's a more detailed overview of the `awk` command:

Basic Syntax:

```
awk 'pattern { action }' file
```

- *pattern*: Specifies a pattern to match in each line of the input.

- *action*: Defines the action to be taken when the pattern is matched. It can include various commands and operations.

Key Concepts:

1. Fields:

- In `awk`, a line is divided into fields by default whitespace (spaces or tabs).

- `$1`, `$2`, etc., represent the first, second, etc., fields of a line.

2. Patterns and Actions:

- A pattern specifies when the associated action should be performed.

- If no pattern is provided, the action is applied to every line.

Example:

```

**awk '/pattern/ {print $1}' file**

```

- This prints the first field of lines that match the specified pattern.

3. Built-in Variables:

- `NR`: Represents the current record (line) number.

- `NF`: Represents the number of fields in the current line.

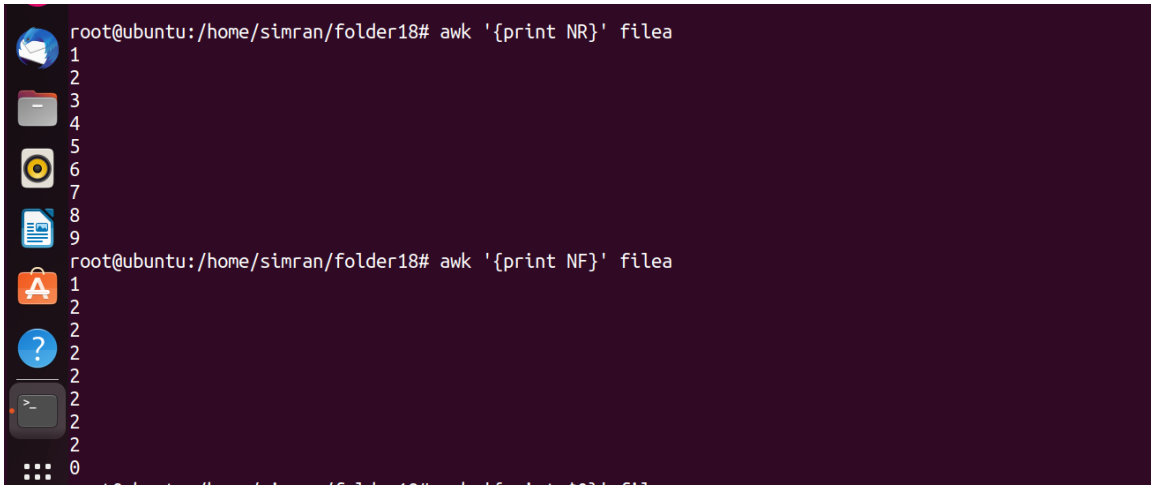
- `$0`: Represents the entire line.

Example:

```

**awk '{print NR, NF, $0}' file**

```



```
root@ubuntu:/home/simran/folder18# awk '{print $0}' filea
Data
Employees     Departments
Sunny            Marketing
rishab           Finance
sneha            Marketing
firoz            Admin
Priya            Account
roy              Marketing
```

- This prints the **record number, number of fields, and the entire line for each line** in the file.

4. User-Defined Variables:

- You can define your own variables in `awk` for more complex processing.

Example:

```

**awk '{total += $1} END {print "Sum:", total}' numbers.txt**

```

- This calculates the sum of the first field in each line and prints the total at the end.

Practice2:

1. Print Specific Columns from a CSV File:
   awk -F',' '{print $1, $3}' filea

```
root@ubuntu:/home/simran/folder18# awk -F',' '{print $1,$3}' filea
Data
Employees     Departments
Sunny            Marketing
rishab           Finance
sneha            Marketing
firoz            Admin
Priya            Account
roy              Marketing
```

2. Filter Lines Based on a Condition:
   awk '$3>30000 {print $1, $2}' filea

```
root@ubuntu:/home/simran/folder18# awk '$3>30000 {print $1, $2}' filea
Employees Departments
rishab Finance
sneha Marketing
firoz Admin
Priya Account
roy Marketing
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees     Departments     Salary
Sunny            Marketing    20000
rishab           Finance      34000
sneha            Marketing    45000
firoz            Admin        40000
Priya            Account      50000
roy              Marketing    60000
```

3. This command separates the columns in the file and give you the view

awk ‘{print $1}’ filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees     Departments
Sunny            Marketing
rishab           Finance
sneha            Marketing
firoz            Admin
Priya            Account
roy              Marketing

root@ubuntu:/home/simran/folder18# awk '{print $1
}' filea
Data
Employees
Sunny
rishab
sneha
firoz
Priya
roy
```

awk ‘{print $2}’ filea

```
root@ubuntu:/home/simran/folder18# awk '{print $2
}' filea

Departments
Marketing
Finance
Marketing
Admin
Account
Marketing

root@ubuntu:/home/simran/folder18# awk '{print $3
}' filea
```

4. Viewing more than one column

awk ‘{print $2, $1}’ filea

```
root@ubuntu:/home/simran/folder18# awk '{print $2, $1}' filea
 Data
Departments Employees
Admin Sunny
Admin rishab
Marketing sneha
Account firoz
Account Priya
Admin roy

root@ubuntu:/home/simran/folder18#
```

5. How to search a word in the file

awk ‘/rishab/ {print $0}’ filea

```
root@ubuntu:/home/simran/folder18# awk '/rishab/{
print $0}' filea
rishab          Admin      34000
root@ubuntu:/home/simran/folder18#
```

6.

If you want to print line number in front of each line

awk '{print NR, $0}' filea

awk '/rishab/ {print NR' $0}' filea

```
root@ubuntu:/home/simran/folder18# awk '{print NR
, $0}' filea
1 Data
2 Employees     Departments      Salary
3 Sunny          Admin     20000
4 rishab                  Admin       34000
5 sneha          Marketing     45000
6 firoz            Account         40000
7 Priya          Account       50000
8 roy            Admin     60000
9
root@ubuntu:/home/simran/folder18# awk '/rishab/
{print NR, $0}' filea
4 rishab                  Admin       34000
root@ubuntu:/home/simran/folder18#
```

7. If want ot search multiple words

awk '/rishab|Priya|Sunny/ {print $0}' filea

```
root@ubuntu:/home/simran/folder18# awk '/rishab|P
riya|Sunny/ {print $0}' filea
Sunny            Admin     20000
rishab           Admin       34000
Priya            Account       50000
root@ubuntu:/home/simran/folder18#
```

8. In second column wherever the n character is present

awk '$2 ~ /n/ {print $0}' filea

```
root@ubuntu:/home/simran/folder18# awk '$2  ~ /n/
 {print $0}' filea
Employees     Departments      Salary
Sunny            Admin     20000
rishab           Admin       34000
sneha            Marketing     45000
firoz            Account         40000
Priya            Account       50000
roy              Admin     60000
root@ubuntu:/home/simran/folder18#
```

9. So far we were working with the file where content is separated by some space in b/w

But we have a .csv file

```
root@ubuntu:/home/simran/folder18# cat fileb.csv
1,seema,rastogi,HR,20000
2,raj,sharma,HR,25000
3,vidisha,rastogi,Accounts,30000
4,shreya,sharma,Admin,35000
5,saanvi,verma,Admin,40000
6,kirat,singh,Accounts,35000
root@ubuntu:/home/simran/folder18#
```

Now the delimiter is , here **awk '{print $2}' fileb.csv** doesn't work. If you want to see data then we need to mention that fields are separated by a **delimiter (,)**

awk -F, '{print $2'} fileb.csv

```
root@ubuntu:/home/simran/folder18# awk '{print $2
}' fileb.csv




root@ubuntu:/home/simran/folder18# awk -F, '{prin
t $2}' fileb.csv
seema
raj
vidisha
shreya
saanvi
kirat
```

10. Now printing the employees whose salary is greater than 25000

awk -F, '$NF>25000 {print $0}' fileb.csv

```
root@ubuntu:/home/simran/folder18# awk -F, '$NF>2
5000{print $0}' fileb.csv
3,vidisha,rastogi,Accounts,30000
4,shreya,sharma,Admin,35000
5,saanvi,verma,Admin,40000
6,kirat,singh,Accounts,35000
root@ubuntu:/home/simran/folder18#
```

11. Incase if file is having multiple delimiters

awk -F[!?] '{print $2}' filec

```
root@ubuntu:/home/simran/folder18# cat filec
Hello There!How are you today?I am good.
root@ubuntu:/home/simran/folder18# awk -F[!?] '{p
rint $2}' filec
How are you today
```

12. To see the file got created after 17th of NOV

ls -ltr | awk '$7>17'

```
-rw-r--r-- 1 root   simran    0 Nov 17 17:53 Filea
-rw-r--r-- 1 root   root      0 Nov 17 17:54 one.html
-rw-r--r-- 1 root   root      0 Nov 17 17:54 one
-rw-r--r-- 1 root   root      9 Nov 17 19:26 filenew
-rw-r--r-- 1 root   root     99 Nov 20 15:26 file.txt
-rwxrwxr-- 1 root   root      9 Nov 21 14:42 file1
-rwxrwx--- 1 root   simran  198 Nov 21 20:18 filea
-rw-r--r-- 1 root   root     25 Nov 21 20:23 filed
-rw-r--r-- 1 root   root    165 Nov 22 13:10 fileb.csv
-rw-r--r-- 1 root   root     41 Nov 22 13:23 filec
root@ubuntu:/home/simran/folder18# ls -ltr | awk '$7>17'
-rw-r--r-- 1 root   root     99 Nov 20 15:26 file.txt
-rwxrwxr-- 1 root   root      9 Nov 21 14:42 file1
-rwxrwx--- 1 root   simran  198 Nov 21 20:18 filea
-rw-r--r-- 1 root   root     25 Nov 21 20:23 filed
-rw-r--r-- 1 root   root    165 Nov 22 13:10 fileb.csv
-rw-r--r-- 1 root   root     41 Nov 22 13:23 filec
root@ubuntu:/home/simran/folder18# ls -ltr | awk '$7<17'
total 40
-rw-r--r-- 1 root   root     24 Nov  5 14:36 xab
drw-rw---- 2 root   root   4096 Nov 16 18:24 folder1
```

5. Built-in Functions:

- `length()`: Returns the length of a string.

- `tolower()`, `toupper()`: Convert strings to lowercase or uppercase.

Example:

```
awk '{if (length($2) > 5) print tolower($2)}' data.txt
```

- This prints the second field in lowercase for lines where its length is greater than 5.

Practical Examples:

1. Modifying the data using built-in functions:

Gsub(“oldword”,”newword”) (gsub is global substitute function)

awk ‘{gsub(“Rishab”,”rudra”); print $0}’ filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000

root@ubuntu:/home/simran/folder18# awk '{gsub("ri
shab","rudra"); print $0}' filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rudra           Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000
```

2. To print the length of character

awk ‘{print length($2)’ filea

awk ‘{print $2, length($2)}’ filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000

root@ubuntu:/home/simran/folder18# awk '{print $2
, length($2)}' filea
 0
Departments 11
Admin 5
Admin 5
Marketing 9
Account 7
Account 7
Admin 5
 0
```

3. if you want to know the position of a word in a line

awk '/Admin/ {print NR, index($0, "Admin")}' filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000

root@ubuntu:/home/simran/folder18# awk '/Admin/ {print NR,
index($0, "Admin")}' filea
3 8
4 9
8 6
root@ubuntu:/home/simran/folder18#
```

4. Printing in Upper and Lowercase

awk '{print tolower($2)}' filea

awk '{print toupper($2)}' filea

```
root@ubuntu:/home/simran/folder18# awk '{print tolower($2)}
' filea

departments
admin
admin
marketing
account
account
admin

root@ubuntu:/home/simran/folder18# awk '{print toupper($2)}
' filea

DEPARTMENTS
ADMIN
ADMIN
MARKETING
ACCOUNT
ACCOUNT
ADMIN
```

AWK Scripting Concepts:

Syntax:

awk

'BEGIN{startaction}

Pattern/condition

END{stopaction}'

Filename

Example:

1. **awk 'BEGIN{print "Data of all employees"} {print $0} END{print "The end of the file"}' filea**

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000

root@ubuntu:/home/simran/folder18# awk  'BEGIN{print "Data
fi all employees:"} {print $0} END{print "The end of the fi
le"}' filea
Data fi all employees:
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000

The end of the file
```

2. How to find the total salary of employees

Here before executing the main block we just declare a variable and stores 0 value in it , then by using NF we are targeting the last column

awk 'BEGIN{sum=0} {sum= sum+$NF} END{print "Sum of Salary " sum}' filea

```
root@ubuntu:/home/simran/folder18# awk 'BEGIN{sum=0} {sum=sum+$NF}
 END{print "sum of Salary " sum}' filea
sum of Salary 249000
root@ubuntu:/home/simran/folder18#
```

3. How to find avg Salary
   - Making a variable calculate and when it goes to next line incrementing it by 1
   - NR>2 to specify from which line you want to calculate the number of employees.
   - Also specify to exclude the any blank line which is present in the file if ($NF>0)

awk 'BEGIN{calculate=0} NR>2 {if($NF>0) calculate++} END{print "Avg of employees Salary is " calculate}' filea

```
root@ubuntu:/home/simran/folder18# awk 'BEGIN{calculate=0} NR>1 {c
alculate++} END{print " The avg of Salary is " calculate}' filea
 The avg of Salary is 8
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees     Departments     Salary
Sunny           Admin     20000
rishab          Admin       34000
sneha           Marketing     45000
firoz           Account         40000
Priya           Account       50000
roy             Admin     60000

root@ubuntu:/home/simran/folder18# awk 'BEGIN{calculate=0} NR>2 {i
f ($NF>0)calculate++} END{print " The avg of Salary is " calculate
}' filea
 The avg of Salary is 6
```

awk 'BEGIN{calculate=0; sum=0'} {if($NF>0)calculate++; sum=sum+$NF} END{print The abg Salary of employees " calculate, sum} ' filea

```
root@ubuntu:/home/simran/folder18# awk 'BEGIN{calculate=0; sum=0}
NR>2 {if ($NF>0)calculate++; sum=sum+$NF} END{print " The avg of S
alary is " calculate, sum}' filea
 The avg of Salary is 6 249000
root@ubuntu:/home/simran/folder18#
```

awk 'BEGIN{calculate=0; sum=0'} {if($NF>0)calculate++; sum=sum+$NF} END{print "The Average salary is" sum/calculate} ' filea

```
root@ubuntu:/home/simran/folder18# awk 'BEGIN{calculate=0; sum=0}
NR>2 {if($NF>1)calculate++; sum=sum+$NF} END{print "The average sa
lary is" sum/calculate}' filea
The average salary is41500
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees     Departments     Salary
Sunny           Admin     20000
rishab          Admin       34000
sneha           Marketing     45000
firoz           Account         40000
Priya           Account       50000
roy             Admin     60000
```

grep Command:

grep stands for "**Global Regular Expression Print**." It is a command-line utility that searches for a pattern in a file or a stream of text and prints lines that contain the pattern.

1. Search for a word in a file:

grep "Admin" filea

grep -i "admin" filea → -I is to ignore the case sensitivity in t searching pattern

2. Search for a word in multiple files:

grep "40000" filea fileb.csv

```
root@ubuntu:/home/simran/folder18# grep "Admin" filea
Sunny           Admin    20000
rishab          Admin      34000
roy             Admin    60000
root@ubuntu:/home/simran/folder18# grep "25000" filea fileb.csv
fileb.csv:2,raj,sharma,HR,25000
root@ubuntu:/home/simran/folder18# cat fileb.csv
1,seema,rastogi,HR,20000
2,raj,sharma,HR,25000
3,vidisha,rastogi,Accounts,30000
4,shreya,sharma,Admin,35000
5,saanvi,verma,Admin,40000
6,kirat,singh,Accounts,35000

root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000
```

```
root@ubuntu:/home/simran/folder18# grep "40000" filea fileb.csv
filea:firoz           Account        40000
fileb.csv:5,saanvi,verma,Admin,40000
```

4. Search recursively in a directory:
   Syntax:

   grep -r "search_term" /path/to/directory

   grep -r “Admin” home/simran/folder18

5. **Case-insensitive search:**
   grep -i “Marketing” filea

```
root@ubuntu:/home/simran/folder18# grep -i "Marketing" filea
sneha           Marketing    45000
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin    20000
rishab          Admin      34000
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
roy             Admin    60000
```

6. Display line **numbers along with matching lines**:
   grep -n “Admin” filea

```
root@ubuntu:/home/simran/folder18# grep -n "Admin" filea
3:Sunny         Admin    20000
4:rishab                Admin      34000
8:roy           Admin    60000
root@ubuntu:/home/simran/folder18#
```

7. Count the number of lines that contain a pattern:
   grep -c “Admin” filea

```
root@ubuntu:/home/simran/folder18# grep -n "Admin" filea
3:Sunny         Admin    20000
4:rishab                Admin      34000
8:roy           Admin    60000
root@ubuntu:/home/simran/folder18# grep -c "Admin" filea
3
```

8. Invert match (display lines that do not contain the pattern):
   grep -v “Admin” filea

```
root@ubuntu:/home/simran/folder18# grep -v "Admin" filea
Data
Employees    Departments     Salary
sneha           Marketing    45000
firoz           Account        40000
Priya           Account      50000
```

How to search multiple patterns:

Grep -e “Admin” -e “Account” filea

```
root@ubuntu:/home/simran/folder18# grep -e "Admin" -e  "Account" file
a
Sunny           Admin     20000
rishab          Admin       34000
firoz           Account         40000
Priya           Account       50000
roy             Admin     60000
root@ubuntu:/home/simran/folder18#
```

egrep Command (grep -E):

- Here Special characters like +, ?, |, (,) have literal meanings unless escaped with a backslash can be used
- To use extended regular expressions (ERE) with grep, you need to escape special characters or use the -E option.

1. Searching multiple patterns with egrep command

egrep “(Admin|Account)” filea

```
root@ubuntu:/home/simran/folder18# egrep "(Admin|Account)" filea
Sunny           Admin     20000
rishab          Admin       34000
firoz           Account         40000
Priya           Account       50000
roy             Admin     60000
root@ubuntu:/home/simran/folder18#
```

2. Word Boundary:
   If you want to search exact word then use -w

   egrep -w “Admin” filea

```
root@ubuntu:/home/simran/folder18# egrep -w "Admin" filea
Sunny           Admin     20000
rishab          Admin       34000
roy             Admin     60000
root@ubuntu:/home/simran/folder18# egrep -w "Adm" filea
root@ubuntu:/home/simran/folder18# egrep  "Adm" filea
Sunny           Admin     20000
rishab          Admin       34000
roy             Admin     60000
root@ubuntu:/home/simran/folder18#
```

3. Matching Lines with Names Starting with "s" or "r"

   egrep “^(s|r)” filea

```
root@ubuntu:/home/simran/folder18# egrep "^(s|r)" filea
rishab          Admin       34000
sneha           Marketing     45000
roy             Admin     60000
root@ubuntu:/home/simran/folder18#
```

4. Matching Lines with Salary Greater Than 40000

```
root@ubuntu:/home/simran/folder18# egrep "[4-5][0-9]{4}"  filea
sneha           Marketing     45000
firoz           Account         40000
Priya           Account       50000
root@ubuntu:/home/simran/folder18#
```

5. Matching Lines with "Account" and Salary Less Than 50000

Egrep “Account.*[0-4][0-5]{4}” filea

```
root@ubuntu:/home/simran/folder18# egrep "Account.* [0-4][0]{4}$" filea
firoz           Account      40000
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
Sunny           Admin        20000
rishab          Admin        34000
sneha           Marketing    45000
firoz           Account      40000
Priya           Account      50000
bhavesh         Account      45000
roy             Admin        60000


root@ubuntu:/home/simran/folder18# egrep "Account.* [0-4][0-5]{4}$" filea
firoz           Account      40000
bhavesh         Account      45000
root@ubuntu:/home/simran/folder18#
```

Sort/Uniq Command:

The **sort** command is used to sort lines of text files. Here are some examples:

1. Sort filea

This command sorts the lines of "filea" in ascending order.

```
root@ubuntu:/home/simran/folder18# sort filea


bhavesh         Account      45000
Data
Employees    Departments     Salary
firoz           Account      40000
Priya           Account      50000
rishab          Admin        34000
roy             Admin        60000
sneha           Marketing    45000
Sunny           Admin        20000
```

2. Sort -r filea

The -r option sorts the lines in descending order.

```
root@ubuntu:/home/simran/folder18# sort -r filea
Sunny           Admin        20000
sneha           Marketing    45000
roy             Admin        60000
rishab          Admin        34000
Priya           Account      50000
firoz           Account      40000
Employees    Departments     Salary
Data
bhavesh         Account      45000
```

3. Sort by a Specific Column:

   Sort -k2 filea

```
root@ubuntu:/home/simran/folder18# sort -k2 filea


Data
firoz           Account      40000
bhavesh         Account      45000
Priya           Account      50000
Sunny           Admin        20000
rishab          Admin        34000
roy             Admin        60000
Employees    Departments     Salary
sneha           Marketing    45000
root@ubuntu:/home/simran/folder18#
```

4. Unique Sorting:
   sort -u filea

```
root@ubuntu:/home/simran/folder18# sort -u filea

bhavesh         Account      45000
Data
David           Admin        45000
Employees    Departments     Salary
firoz           Account      40000
Priya           Account      50000
rishab          Admin        34000
roy             Admin        60000
sneha           Marketing    45000
Sunny           Admin        20000
root@ubuntu:/home/simran/folder18#
```

uniq Command:

The **uniq** command is used to filter adjacent matching lines from a sorted file. It is often used in combination with sort. Here are some examples:

1. Uniq filea

```
root@ubuntu:/home/simran/folder18# uniq filea
Data
Employees    Departments     Salary
SunnyAdmin20000
rishabAdmin34000
snehaMarketing45000
firozAccount40000
PriyaAccount50000
bhaveshAccount45000
royAdmin60000
DavidAdmin45000
root@ubuntu:/home/simran/folder18#
```

2. Counting Duplicate Lines
   Uniq -c filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
SunnyAdmin20000
rishabAdmin34000
snehaMarketing45000
firozAccount40000
PriyaAccount50000
bhaveshAccount45000
royAdmin60000
royAdmin60000
DavidAdmin45000
root@ubuntu:/home/simran/folder18# uniq -c filea
      1 Data
      1 Employees    Departments     Salary
      1 SunnyAdmin20000
      1 rishabAdmin34000
      1 snehaMarketing45000
      1 firozAccount40000
      1 PriyaAccount50000
      1 bhaveshAccount45000
      2 royAdmin60000
      1 DavidAdmin45000
```

3. Display Only Duplicate Lines

Uniq -d filea

```
root@ubuntu:/home/simran/folder18# uniq -d filea
royAdmin60000
root@ubuntu:/home/simran/folder18#
```

4. sort filea | uniq

```
root@ubuntu:/home/simran/folder18# sort filea | uniq
bhaveshAccount45000
Data
DavidAdmin45000
Employees    Departments     Salary
firozAccount40000
PriyaAccount50000
rishabAdmin34000
royAdmin60000
snehaMarketing45000
SunnyAdmin20000
```

5. If you want to count the occurrences of each unique line, you can use the -c option with uniq:

```
root@ubuntu:/home/simran/folder18# sort filea |uniq -c
      1 bhaveshAccount45000
      1 Data
      1 DavidAdmin45000
      1 Employees    Departments     Salary
      1 firozAccount40000
      1 PriyaAccount50000
      1 rishabAdmin34000
      2 royAdmin60000
      1 snehaMarketing45000
      1 SunnyAdmin20000
root@ubuntu:/home/simran/folder18#
```

Diff & Cmp Command:

The **diff** and **cmp** commands in Linux are used to compare files and identify the differences between them

1. Diff filea fileb.csv

```
root@ubuntu:/home/simran/folder18# diff filea fileb.csv
1,11c1,7
< Data
< Employees    Departments     Salary
< SunnyAdmin20000
< rishabAdmin34000
< snehaMarketing45000
< firozAccount40000
< PriyaAccount50000
< bhaveshAccount45000
< royAdmin60000
< royAdmin60000
< DavidAdmin45000
---
> 1,seema,rastogi,HR,20000
> 2,raj,sharma,HR,25000
> 3,vidisha,rastogi,Accounts,30000
> 4,shreya,sharma,Admin,35000
> 5,saanvi,verma,Admin,40000
> 6,kirat,singh,Accounts,35000
>
```

2. **-u or --unified:** Provides a unified diff, showing several lines of context

Diff -u filea fileb.csv

```
root@ubuntu:/home/simran/folder18# diff -u filea fileb.csv
--- filea       2023-11-23 14:19:25.580480697 +0530
+++ fileb.csv   2023-11-22 13:10:57.118087087 +0530
@@ -1,11 +1,7 @@
-Data
-Employees    Departments     Salary
-SunnyAdmin20000
-rishabAdmin34000
-snehaMarketing45000
-firozAccount40000
-PriyaAccount50000
-bhaveshAccount45000
-royAdmin60000
-royAdmin60000
-DavidAdmin45000
+1,seema,rastogi,HR,20000
+2,raj,sharma,HR,25000
+3,vidisha,rastogi,Accounts,30000
+4,shreya,sharma,Admin,35000
+5,saanvi,verma,Admin,40000
+6,kirat,singh,Accounts,35000
```

3. **-c or --context**: Provides a context diff.

```
root@ubuntu:/home/simran/folder18# diff -c filea fileb.csv
*** filea       2023-11-23 14:19:25.580480697 +0530
--- fileb.csv   2023-11-22 13:10:57.118087087 +0530
***************
*** 1,11 ****
! Data
! Employees    Departments     Salary
! SunnyAdmin20000
! rishabAdmin34000
! snehaMarketing45000
! firozAccount40000
! PriyaAccount50000
! bhaveshAccount45000
! royAdmin60000
! royAdmin60000
! DavidAdmin45000
--- 1,7 ----
! 1,seema,rastogi,HR,20000
! 2,raj,sharma,HR,25000
! 3,vidisha,rastogi,Accounts,30000
! 4,shreya,sharma,Admin,35000
! 5,saanvi,verma,Admin,40000
! 6,kirat,singh,Accounts,35000
!
```

cmp Command:

The **cmp** command compares two files byte by byte and reports the first mismatch. It's often used to quickly check if two files are identical.

1. Create two identical files

```
root@ubuntu:/home/simran/folder18# echo "This is new file!">file1.txt
root@ubuntu:/home/simran/folder18# echo "This is new file!">file2.txt
root@ubuntu:/home/simran/folder18# ls
file1      file2.txt  Filea      filec    filer     folder2
file1.txt  file33     fileb      filed    file.txt  one.html
File21     filea      fileb.csv  filenew  folder1   xab
root@ubuntu:/home/simran/folder18#
```

If the files are identical, the **cmp** command produces no output. In this case, both files are the same, so you won't see anything as the output.

```
root@ubuntu:/home/simran/folder18# ls
file1      file2.txt  Filea      filec    filer     folder2
file1.txt  file33     fileb      filed    file.txt  one.html
File21     filea      fileb.csv  filenew  folder1   xab
root@ubuntu:/home/simran/folder18# cmp file1.txt file2.txt
root@ubuntu:/home/simran/folder18#
```

2. Create two differ files

```
root@ubuntu:/home/simran/folder18# cat file1.txt
This is new file!
root@ubuntu:/home/simran/folder18# cat file2.txt
This is an old file.
root@ubuntu:/home/simran/folder18#
```

```
root@ubuntu:/home/simran/folder18# cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 9, line 1
root@ubuntu:/home/simran/folder18#
```

This output indicates that the files differ at byte 9, which is the first byte position where the content differs.

3. **Verbose Output**

To get more information about the differences, use the **-l** option:

```
root@ubuntu:/home/simran/folder18# cmp -l file1.txt file2.txt
 9 156 141
10 145 156
11 167  40
12  40 157
13 146 154
14 151 144
15 154  40
16 145 146
17  41 151
18  12 154
cmp: EOF on file1.txt after byte 18
root@ubuntu:/home/simran/folder18#
```

This output indicates that at byte 9, file1.txt has the value 156, while file2.txt has the value 141.

Wc Command:

The **wc** command in Linux is used to count the number of words, lines, and bytes in a file or text input.

1. Wc filea

```
root@ubuntu:/home/simran/folder18# wc filea
 11  13 194 filea
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
SunnyAdmin20000
rishabAdmin34000
snehaMarketing45000
firozAccount40000
PriyaAccount50000
bhaveshAccount45000
royAdmin60000
royAdmin60000
DavidAdmin45000
root@ubuntu:/home/simran/folder18#
```

2. Counting the lines only
   Wc -l filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
SunnyAdmin20000
rishabAdmin34000
snehaMarketing45000
firozAccount40000
PriyaAccount50000
bhaveshAccount45000
royAdmin60000
royAdmin60000
DavidAdmin45000
root@ubuntu:/home/simran/folder18# wc -l filea
11 filea
```

3. Counting words only

   Wc -w filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
SunnyAdmin20000
rishabAdmin34000
snehaMarketing45000
firozAccount40000
PriyaAccount50000
bhaveshAccount45000
royAdmin60000
royAdmin60000
DavidAdmin45000
root@ubuntu:/home/simran/folder18# wc -l filea
11 filea
root@ubuntu:/home/simran/folder18# wc -w filea
13 filea
```

4. Counting bytes only

   Wc -c filea

```
root@ubuntu:/home/simran/folder18# cat filea
Data
Employees    Departments     Salary
SunnyAdmin20000
rishabAdmin34000
snehaMarketing45000
firozAccount40000
PriyaAccount50000
bhaveshAccount45000
royAdmin60000
royAdmin60000
DavidAdmin45000
root@ubuntu:/home/simran/folder18# wc -c filea
194 filea
```

5. Counting multiple files

Wc file1.txt file2.txt

```
194 filea
root@ubuntu:/home/simran/folder18# wc file1.txt file2.txt
 1  4 18 file1.txt
 1  5 21 file2.txt
 2  9 39 total
root@ubuntu:/home/simran/folder18#
```

6. Reading from Standard Input

   You can also use wc to read from standard input by providing data through a pipe **(|)**:

```
root@ubuntu:/home/simran/folder18# echo "Hello There!" |wc
       1       2      13
root@ubuntu:/home/simran/folder18#
```

1 line 2 words 13 bytes