Practicals on OS
UNIT III
FILTERS

cut Command

cut command is useful for selecting a specific column of a file. It is used to cut a specific sections by byte position, character, and field and writes them to standard output. It cuts a line and extracts the text data. It is necessary to pass an argument with it; otherwise, it will throw an error message.

To cut a specific section, it is necessary to specify the delimiter. A delimiter will decide how the sections are separated in a text file. Delimiters can be a space (' '), a hyphen (-), a slash (/), or anything else. After '-f' option, the column number is mentioned.

Syntax

cut OPTION... [FILE]...

Options:

-b, --bytes=LIST: It is used to cut a specific section by bytes.

-c, --characters=LIST: It is used to select the specified characters.

-d, --delimiter=DELIM: It is used to cut a specific section by a delimiter.

-f, --fields=LIST: It is used to select the specific fields. It also prints any line that does not contain any delimiter character, unless the -s option is specified.

-n: It is used to ignore any option.

--complement: It is used to complement the set of selected bytes, characters or fields

-s, --only-delimited: It is used to not print lines that do not have delimiters.

--output-delimiter=STRING: This option is specified to use a STRING as an output delimiter; The default is to use "input delimiter".

-z, --zero-terminated: It is used if line delimiter is NUL, not newline.

--help: It is used to display the help manual.

--version: It is used to display the version information.

DEMO

Using Hyphen (-) As Delimiter

Syntax:- cut -d- -f(columnNumber) <fileName>

student@fcait-ug-41:~/Desktop$ cat > marks.txt
alex-50
alen-70
john-75
carry-85
celena-90
justin-80

student@fcait-ug-41:~/Desktop$ cut -d- -f2 marks.txt

student@fcait-ug-41:~/Desktop$ cut -d- -f1 marks.txt

Using Space As Delimiter

Syntax:- cut -d ' ' -f(columnNumber) <fileName>

student@fcait-ug-41:~/Desktop$ cat > exam.txt
Apple is red
Mango is yellow
Dress color is red
Red color suits all

student@fcait-ug-41:~/Desktop$ cut -d ' ' -f2 exam.txt

student@fcait-ug-41:~/Desktop$ cut -d ' ' -f4 exam.txt

cut by Byte

Syntax:- cut -b <byte number> <file name>

student@fcait-ug-41:~/Desktop$ cut -b 2 exam.txt

cut by Character

Syntax:- cut -c < characters> <file name>

student@fcait-ug-41:~/Desktop$ cut -c 1,6 exam.txt
A
M
D
Ro
student@fcait-ug-41:~/Desktop$ cut -c 1-3 exam.txt
App
Man
Dre

Red
cut by Complement Pattern

Syntax:- cut --complement < complement pattern> <file name>

student@fcait-ug-41:~/Desktop$ cut --complement -c 1 exam.txt
pple is red
ango is yellow
ress color is red
ed color suits all

student@fcait-ug-41:~/Desktop$ cut --complement -c 2 exam.txt
Aple is red
Mngo is yellow
Dess color is red
Rd color suits all

student@fcait-ug-41:~/Desktop$ cut --complement -c 2-4 exam.txt
Ae is red
Mo is yellow
Ds color is red
Rcolor suits all

paste Command

It is used to join files horizontally (parallel merging) by outputting lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output. When no file is specified, or put dash ("-") instead of file name, paste reads from standard input and gives output as it is until a interrupt command [Ctrl-c] is given. Syntax:

paste [OPTION]... [FILES]...

student@fcait-ug-41:~/Desktop$ cat > state
Arunachal Pradesh
Assam
Andhra Pradesh
Bihar
Chhattisgrah

student@fcait-ug-41:~/Desktop$ cat > capital
Itanagar
Dispur
Hyderabad
Patna
Raipur

student@fcait-ug-41:~/Desktop$ cat > number
1
2
3
4

Without any option paste merges the files in parallel. The paste command writes corresponding lines from the files with tab as a deliminator on the terminal.

student@fcait-ug-41:~/Desktop$ paste number state capital

Options:-

1. -d (delimiter): Paste command uses the tab delimiter by default for merging the files. The delimiter can be changed to any other character by using the -d option. If more than one character is specified as delimiter then paste uses it in a circular fashion for each file line separation.

Only one character is specified
student@fcait-ug-41:~/Desktop$ paste -d "|" number state capital

More than one character is specified
student@fcait-ug-41:~/Desktop$ paste -d "|," number state capital

2. -s (serial): We can merge the files in sequentially manner using the -s option. It reads all the lines from a single file and merges all these lines into a single line with each line separated by tab. And these single lines are separated by newline.

student@fcait-ug-41:~/Desktop$ paste -s number state capital
student@fcait-ug-41:~/Desktop$ paste -s -d ":" number state capital

Applications of Paste Command

1. Combining N consecutive lines: The paste command can also be used to merge N consecutive lines from a file into a single line. Here N can be specified by specifying number hyphens(-) after paste.

With 2 hyphens
student@fcait-ug-41:~/Desktop$ cat capital | paste - -

With 3 hyphens
student@fcait-ug-41:~/Desktop$ paste - - - < capital

2. Combination with other commands: Even though paste require at least two files for concatenating lines, but data from one file can be given from shell.

student@fcait-ug-41:~/Desktop$ cut -d " " -f 1 state | paste number -

join Command

The join command in UNIX is a command line utility for joining lines of two files on a common field.

Syntax:

$join [OPTION] FILE1 FILE2

EXAMPLE

student@fcait-ug-41:~/Desktop$ cat > file1.txt

student@fcait-ug-41:~/Desktop$ cat > file2.txt

Now, in order to combine two files the files must have some common field. In this case, we have the numbering 1, 2... as the common field in both the files.

student@fcait-ug-41:~/Desktop$ join file1.txt file2.txt

student@fcait-ug-41:~/Desktop$ join file1.txt file2.txt > newjoinfile.txt

student@fcait-ug-41:~/Desktop$ cat newjoinfile.txt

Options

1. using -a FILENUM option : Also, print unpairable lines from file FILENUM, where FILENUM is 1 or 2, corresponding to FILE1 or FILE2.

student@fcait-ug-41:~/Desktop$ pico file1.txt
Add 1 more line ---------- 5 DEEPAK

student@fcait-ug-41:~/Desktop$ join file1.txt file2.txt -a 1

2. using -v option : Like -a FILENUM, but suppress joined output lines.

student@fcait-ug-41:~/Desktop$ join file1.txt file2.txt -v 1

3. using -1, -2 and -j option : Now, if you want the second field of either file or both the files to be the common field for join, you can do this by using the -1 and -2 command line options. The -1 and -2 here represents he first and second file and these options requires a numeric argument that refers to the joining field for the corresponding file.

student@fcait-ug-41:~/Desktop$ join -1 1 -2 1 file1.txt file2.txt

student@fcait-ug-41:~/Desktop$ join -j1 file1.txt file2.txt

4. using -i option : Ignore differences in case when comparing fields.

student@fcait-ug-41:~/Desktop$ cat > f1.txt
A AAYUSH
B APAAR
C HEMANT
D KARTIK

student@fcait-ug-41:~/Desktop$ cat > f2.txt
a 101
b 102
c 103
d 104

student@fcait-ug-41:~/Desktop$ join -i file1.txt file2.txt

5. using -t option : Use CHAR as input and output field separator.

student@fcait-ug-41:~/Desktop$ pico f1.txt
1, AAYUSH
2, APAAR
3, HEMANT
4, KARTIK

student@fcait-ug-41:~/Desktop$ pico f2.txt
1, 101
2, 102
3, 103
4, 104

student@fcait-ug-41:~/Desktop$ join -t, f1.txt f2.txt

fold Command

The fold command in Linux wraps each line in an input file to fit a specified width and prints it to the standard output. By default, it wraps lines at a maximum width of 80 columns, which is configurable. To fold input using the fold command pass a file or standard input to the command.

Syntax:

fold [OPTION] [FILE]

Example:

tudent@fcait-ug-41:~/Desktop$ cat > file.txt
Linux is a community of open-source Unix like operating systems that are based on the Linux Kernel. It was initially released by Linus Torvalds on September 17, 1991. It is a free and open-source operating system and the source code can be modified and distributed to anyone commercially or noncommercially under the GNU General Public License.
Initially, Linux was created for personal computers and gradually it was used in other machines like servers, mainframe computers, supercomputers, etc. Nowadays, Linux is also used in embedded systems like routers, automation controls, televisions, digital video recorders, video game consoles, smartwatches, etc. The biggest success of Linux is Android(operating system) it is based on the Linux kernel that is running on smartphones and tablets. Due to android Linux has the largest installed base of all general-purpose operating systems. Linux is generally packaged in a Linux distribution.

student@fcait-ug-41:~/Desktop$ fold file.txt

- Wrapping Lines with the `-w` option in `fold` command:
  By using this option in the fold command, we can limit the width by the number of columns. Using this command, we change the column width from the default width 80.

student@fcait-ug-41:~/Desktop$ fold -w 50 file.txt

- Handling Nonprinting Characters with the `-b` option:
  This option of fold command is used to limit the width of the output by the number of bytes rather than the number of columns. By using this we can enforce the width of the output to the number of bytes.

```
student@fcait-ug-41:~/Desktop$ fold -b40 file.txt
```

- Truncating Lines with the `-s` option:
  This option is used to break the lines on spaces so that words are not broken. If a segment of the line contains a blank character within the first width column positions, break the line after the last such blank character meeting the width constraints.

```
student@fcait-ug-41:~/Desktop$ fold -w60 -s file.txt
```

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.

To sort:-

```
student@fcait-ug-41:~/Desktop$ cat > file.txt
abhishek
chitransh
satish
rajan
naveen
divyam
harsh

student@fcait-ug-41:~/Desktop$ sort file.txt
```

Sort function with mix file :-

```
student@fcait-ug-41:~/Desktop$ cat > mix.txt
abc
apple
BALL
Abc
bat
```

Options:

1. -o Option: to write the output to a new file. Using the -o option is functionally the same as redirecting the output to a file.

```
student@fcait-ug-41:~/Desktop$ sort -o output.txt file.txt
student@fcait-ug-41:~/Desktop$ cat output.txt
```

2. -r Option: Sorting In Reverse Order.

```
student@fcait-ug-41:~/Desktop$ sort -r file.txt
```

3. -n Option: To sort a file numerically.

```
student@fcait-ug-41:~/Desktop$ cat > file1.txt
50
39
15
89
200

student@fcait-ug-41:~/Desktop$ sort -n file1.txt
```

4. -nr option: To sort a file with numeric data in reverse order.

```
student@fcait-ug-41:~/Desktop$ sort -nr file1.txt
```

5. -k Option: Sorting a table on the basis of any column number by using -k option.

Use the -k option to sort on a certain column. For example, use “-k 2” to sort on the second column.

```
student@fcait-ug-41:~/Desktop$ sort -k 2 employee.txt

student@fcait-ug-41:~/Desktop$ sort -k 1 employee.txt
```

6. -c option: This option is used to check if the file given is already sorted or not & checks if a file is already sorted pass the -c option to sort.

```
student@fcait-ug-41:~/Desktop$ sort -c file.txt
```

Note : If there is no output then the file is considered to be already sorted

7. -u option: To sort and remove duplicates

```
student@fcait-ug-41:~/Desktop$ cat > car.txt
Audi
BMW
Cadillac
BMW
Dodge

student@fcait-ug-41:~/Desktop$ sort -u car.txt
```

8. -M Option: To sort by month

```
student@fcait-ug-41:~/Desktop$ cat > months.txt
February
January
March
August
```

```
student@fcait-ug-41:~/Desktop$ sort -M months.txt
```

tr Command

The tr command is a UNIX 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.

Syntax :

$ tr [OPTION] SET1 [SET2]

Options
-c : complements the set of characters in string.i.e., operations apply to characters not in the given set
-d : delete characters in the first set from the output.
-s : replaces repeated characters listed in the set1 with single occurrence
-t : truncates set1.

1. How to convert lower case characters to upper case.

```
student@fcait-ug-41:~/Desktop$ cat > gls.txt
WELCOME to
gls

student@fcait-ug-41:~/Desktop$ cat gls.txt | tr [a-z] [A-Z]
                          OR
student@fcait-ug-41:~/Desktop$ cat gls.txt | tr [:lower:] [:upper:]
                          OR
student@fcait-ug-41:~/Desktop$ tr [:lower:] [:upper:] < gls.txt
```

2. How to translate white-space characters to tabs.

```
echo "Welcome To GLS" | tr [:space:] "\t"
                  OR
tr [:space:] "\t" <<< "Welcome To GLS"
```

3. How to translate braces into parenthesis.

```
student@fcait-ug-41:~/Desktop$ cat > gls.txt
{WELCOME TO}
GLS

student@fcait-ug-41:~/Desktop$ tr "{}" "()" < gls.txt > new.txt

student@fcait-ug-41:~/Desktop$ cat new.txt
```

4. How to squeeze a sequence of repetitive characters using -s option.

```
student@fcait-ug-41:~/Desktop$ echo "Welcome    To    GLS" | tr -s " "
Welcome To GLS
                                OR
student@fcait-ug-41:~/Desktop$ tr -s " " <<< "Welcome    To    GLS"
Welcome To GLS
```

5. How to delete specified characters using -d option.

```
student@fcait-ug-41:~/Desktop$ echo "Welcome To GLS" | tr -d W
elcome To GLS
student@fcait-ug-41:~/Desktop$ tr -d W <<< "Welcome to GLS"
elcome to GLS
```

6. To remove all the digits from the string, you can use

```
echo "my ID is 73535" | tr -d [:digit:]
                or

tr -d [:digit:] <<< "my ID is 73535"
```

7. How to complement the sets using -c option

```
echo "my ID is 73535" | tr -cd [:digit:]
                or

tr -cd [:digit:] <<< "my ID is 73535"
```

uniq Command

uniq is the tool that helps to detect the adjacent duplicate lines and also deletes the duplicate lines. uniq filters out the adjacent matching lines from the input file(that is required as an argument) and writes the filtered data to the output file.

Syntax of uniq Command :

$uniq [OPTION] [INPUT[OUTPUT]]

```
student@fcait-ug-41:~/Desktop$ cat > music.txt
I love music.
I love music.
I love music.

Thanks.

student@fcait-ug-41:~/Desktop$ uniq music.txt
```

Note: uniq isn’t able to detect the duplicate lines unless they are adjacent to each other. The content in the file must be therefore sorted before using uniq or you can simply use sort -u instead of uniq command.

1. Using -c option : It tells the number of times a line was repeated.

```
student@fcait-ug-41:~/Desktop$ uniq -c music.txt
```

2. Using -d option : It only prints the repeated lines.

```
student@fcait-ug-41:~/Desktop$ uniq -d music.txt
```

3. Using -D option : It also prints only duplicate lines but not one per group.

```
student@fcait-ug-41:~/Desktop$ uniq -D music.txt
```

4. Using -u option : It prints only the unique lines.

```
student@fcait-ug-41:~/Desktop$ uniq -u music.txt
```

5. Using -f N option : This allows the N fields to be skipped while comparing the uniqueness of the lines. This option is helpful when the lines are numbered as shown in the example below:

```
student@fcait-ug-41:~/Desktop$ cat > f1.txt
1. I love music.
2. I love music.
3. I love music of Kartik.
4. I love music of Kartik.

student@fcait-ug-41:~/Desktop$ uniq -f 2 f1.txt
```

6. Using -s N option : This is similar to -f N option but it skips N characters but not N fields.

```
student@fcait-ug-41:~/Desktop$ cat > f2.txt
#%@I love music.
^&(I love music.
*-!@thanks.
#%@!thanks.

student@fcait-ug-41:~/Desktop$ uniq -s 3 f2.txt
```

7. Using -w option : Similar to the way of skipping characters, we can also ask uniq to limit the comparison to a set number of characters. For this, -w command-line option is used.

```
student@fcait-ug-41:~/Desktop$ cat > f3.txt
How it is possible?
How it can be done?
How to use it?
```

student@fcait-ug-41:~/Desktop$ uniq -w 3 f3.txt

8. Using -i option : It is used to make the comparison case-insensitive.

student@fcait-ug-41:~/Desktop$ cat > f4.txt
I LOVE MUSIC
i love music
THANKS

student@fcait-ug-41:~/Desktop$ uniq -i f4.txt

head Command

The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

Syntax:

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

student@fcait-ug-41:~/Desktop$ cat > state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana

Tripura
Uttar Pradesh
Uttarakhand
West Bengal

student@fcait-ug-41:~/Desktop$ head state.txt

Options:-

1. -n num: Prints the first ‘num’ lines instead of first 10 lines. num is mandatory to be specified in command otherwise it displays an error.

student@fcait-ug-41:~/Desktop$ head -n 5 state.txt

2. -c num: Prints the first ‘num’ bytes from the file specified. Newline count as a single character, so if head prints out a newline, it will count it as a byte. num is mandatory to be specified in command otherwise displays an error.

student@fcait-ug-41:~/Desktop$ head -c 10 state.txt

3. -q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.

student@fcait-ug-41:~/Desktop$ cat > capital.txt
Hyderabad
Itanagar
Dispur
Patna
Raipur
Panaji
Gandhinagar
Chandigarh
Shimla
Srinagar

Without using -q option
student@fcait-ug-41:~/Desktop$ head state.txt capital.txt

With using -q option
student@fcait-ug-41:~/Desktop$ head -q state.txt capital.txt

4. -v: By using this option, data from the specified file is always preceded by its file name.

student@fcait-ug-41:~/Desktop$ head -v state.txt

5. Print line between M and N lines(M>N):

For this purpose, we use the head, tail, and pipeline(|) commands. The command is: head -M file_name | tail +N since the head command takes first M lines and from M lines tail command cuts lines starting from +N till the end, we can also use head -M file_name | tail +(M-N+1) command since the head command takes first M lines and from M lines tail command cuts (M-N+1) lines starting from the end. Let say from the state.txt file we have to print lines between 10 and 20.

student@fcait-ug-41:~/Desktop$ head -n 20 state.txt | tail -10

tail Command

The tail command, as the name implies, print the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.

Syntax:

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

student@fcait-ug-41:~/Desktop$ tail state.txt

Options:-

1. -n num: Prints the last ‘num’ lines instead of last 10 lines.

student@fcait-ug-41:~/Desktop$ tail -n 3 state.txt
OR
student@fcait-ug-41:~/Desktop$ tail -3 state.txt

Tail command also comes with an ‘+’ option which is not present in the head command. With this option tail command prints the data starting from specified line number of the file instead of end.

student@fcait-ug-41:~/Desktop$ tail +25 state.txt

2. -c num: Prints the last ‘num’ bytes from the file specified. Newline count as a single character, so if tail prints out a newline, it will count it as a byte.
Note: Without positive or negative sign before num, command will display the last num bytes from the file specified.

With negative:-
student@fcait-ug-41:~/Desktop$ tail -c -7 state.txt
OR
student@fcait-ug-41:~/Desktop$ tail -c 7 state.txt

With positive:-
student@fcait-ug-41:~/Desktop$ tail -c +263 state.txt

3. -q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.

Without -q option:-
student@fcait-ug-41:~/Desktop$ tail state.txt capital.txt

With -q option:-
student@fcait-ug-41:~/Desktop$ tail -q state.txt capital.txt

4. -f: This option is mainly used by system administration to monitor the growth of the log files written by many Unix program as they are running. This option shows the last ten lines of a file and will update when new lines are added.

5. -v: By using this option, data from the specified file is always preceded by its file name.

student@fcait-ug-41:~/Desktop$ tail -v state.txt

6. –version: This option is used to display the version of tail which is currently running on your system.

student@fcait-ug-41:~/Desktop$ tail –version

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 global search for regular expression and print out).

Syntax:

grep [options] pattern [files]

student@fcait-ug-41:~/Desktop$ cat > case.txt
unix is great os. unix was developed in Bell labs.
learn operating system.
Unix linux which one you choose.
uNix is easy to learn.unix is a multiuser os.Learn unix .unix is a powerful.\

1. Case insensitive search : The -i option enables to search for a string case insensitively in the given file. It matches the words like “UNIX”, “Unix”, “unix”.

student@fcait-ug-41:~/Desktop$ grep -i "UNix" case.txt

2. Displaying the count of number of matches : We can find the number of lines that matches the given string/pattern

student@fcait-ug-41:~/Desktop$ grep -c "unix" case.txt

3. Display the file names that matches the pattern : We can just display the files that contains the given string/pattern.

```
student@fcait-ug-41:~/Desktop$ grep -l "unix" *
                              OR
student@fcait-ug-41:~/Desktop$ grep -l "unix" f1.txt f2.txt f3.txt case.txt
```

4. Checking for the whole words in a file : By default, grep matches the given string/pattern even if it is found as a substring in a file. The -w option to grep makes it match only the whole words.

```
student@fcait-ug-41:~/Desktop$ grep -w "unix" case.txt
```

5. Displaying only the matched pattern : By default, grep displays the entire line which has the matched string. We can make the grep to display only the matched string by using the -o option.

```
student@fcait-ug-41:~/Desktop$ grep -o "unix" case.txt
```

6. Show line number while displaying the output using grep -n : To show the line number of file with the line matched.

```
student@fcait-ug-41:~/Desktop$ grep -n "unix" case.txt
```

7. Inverting the pattern match : You can display the lines that are not matched with the specified search string pattern using the -v option.

```
 student@fcait-ug-41:~/Desktop$ grep -v "unix" case.txt
```

8. Matching the lines that start with a string : The ^ regular expression pattern specifies the start of a line. This can be used in grep to match the lines which start with the given string or pattern.

```
student@fcait-ug-41:~/Desktop$ grep "^unix" case.txt
```

9. Matching the lines that end with a string : The $ regular expression pattern specifies the end of a line. This can be used in grep to match the lines which end with the given string or pattern.

```
student@fcait-ug-41:~/Desktop$ grep "os$" case.txt
```

10.Specifies expression with -e option. Can use multiple times :

```
student@fcait-ug-41:~/Desktop$ grep -e "Unix" -e "os" -e "linux" case.txt
```

11. Print n specific lines from a file: -A prints the searched line and n lines after the result, -B prints the searched line and n lines before the result, and -C prints the searched line and n lines after and before the result.

Syntax:

$grep -A[NumberOfLines(n)] [search] [file]

$grep -B[NumberOfLines(n)] [search] [file]

$grep -C[NumberOfLines(n)] [search] [file]

student@fcait-ug-41:~/Desktop$ grep -A1 learn case.txt

sed Command

SED command in UNIX stands for stream editor and it can perform lots of functions on file like searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace.

Syntax:

sed OPTIONS... [SCRIPT] [INPUTFILE...]

1. Replacing or substituting string : Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

student@fcait-ug-41:~/Desktop$ sed 's/unix/linux/' case.txt

2. Replacing the nth occurrence of a pattern in a line : Use the /1, /2 etc flags to replace the first, second occurrence of a pattern in a line. The below command replaces the second occurrence of the word “unix” with “linux” in a line.

student@fcait-ug-41:~/Desktop$ sed 's/unix/linux/2' case.txt

3. Replacing all the occurrence of the pattern in a line : The substitute flag /g (global replacement) specifies the sed command to replace all the occurrences of the string in the line.

student@fcait-ug-41:~/Desktop$ sed 's/unix/linux/g' case.txt

4. Replacing from nth occurrence to all occurrences in a line : Use the combination of /1, /2 etc and /g to replace all the patterns from the nth occurrence of a pattern in a line. The following sed command replaces the third, fourth, fifth… “unix” word with “linux” word in a line.

student@fcait-ug-41:~/Desktop$ sed 's/unix/linux/3g' case.txt

5. Parenthesize first character of each word : This sed example prints the first character of every word in parenthesis.

student@fcait-ug-41:~/Desktop$ echo "Welcome To The GLS University" | sed 's/\(\b[A-Z]\)/\(\1\)/g'

6. Replacing string on a specific line number : You can restrict the sed command to replace the string on a specific line number. An example is

student@fcait-ug-41:~/Desktop$ sed '3 s/Unix/linux/' case.txt

7. Duplicating the replaced line with /p flag : The /p print flag prints the replaced line twice on the terminal. If a line does not have the search pattern and is not replaced, then the /p prints that line only once.

student@fcait-ug-41:~/Desktop$ sed 's/Unix/linux/p' case.txt

8. Printing only the replaced lines : Use the -n option along with the /p print flag to display only the replaced lines. Here the -n option suppresses the duplicate rows generated by the /p flag and prints the replaced lines only one time.

student@fcait-ug-41:~/Desktop$ sed -n 's/Unix/linux/p' case.txt

9. Replacing string on a range of lines : You can specify a range of line numbers to the sed command for replacing a string.

student@fcait-ug-41:~/Desktop$ sed '1,3 s/Unix/linux/' case.txt

10. Deleting lines from a particular file : SED command can also be used for deleting lines from a particular file. SED command is used for performing deletion operation without even opening the file

Examples:
1. To Delete a particular line say n in this example
Syntax:
$ sed 'nd' filename.txt

Example:
$ sed '5d' filename.txt

2. To Delete a last line

Syntax:
$ sed '$d' filename.txt
3. To Delete line from range x to y

Syntax:
$ sed 'x,yd' filename.txt
Example:
$ sed '3,6d' filename.txt
4. To Delete from nth to last line

Syntax:
$ sed 'nth,$d' filename.txt
Example:

$ sed '12,$d' filename.txt
5. To Delete pattern matching line

Syntax:
$ sed '/pattern/d' filename.txt
Example:
$ sed '/abc/d' filename.txt

awk Command

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. Awk is mostly used for pattern scanning and processing. It searches one or more files to see if they contain lines that matches with the specified patterns and then perform the associated actions.

Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan.

1. AWK Operations:
(a) Scans a file line by line
(b) Splits each input line into fields
(c) Compares input line/fields to pattern
(d) Performs action(s) on matched lines

2. Useful For:
(a) Transform data files
(b) Produce formatted reports

3. Programming Constructs:
(a) Format output lines
(b) Arithmetic and string operations
(c) Conditionals and loops

Syntax:

awk options 'selection _criteria {action }' input-file > output-file
Options:

-f program-file : Reads the AWK program source from the file
program-file, instead of from the
first command line argument.
-F fs : Use fs for the input field separator

EXAMPLE

student@fcait-ug-41:~/Desktop$ cat > employee.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000

amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000

1. Default behavior of Awk: By default Awk prints every line of data from the specified file.

student@fcait-ug-41:~/Desktop$ awk '{print}' employee.txt

2. Print the lines which match the given pattern.

student@fcait-ug-41:~/Desktop$ awk '/manager/ {print}' employee.txt

3. Splitting a Line Into Fields : For each record i.e line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

student@fcait-ug-41:~/Desktop$ awk '{print $1,$4}' employee.txt

Built-In Variables In Awk

Awk's built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

NR: NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file.
NF: NF command keeps a count of the number of fields within the current input record.
FS: FS command contains the field separator character which is used to divide fields on the input line. The default is "white space", meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator.
RS: RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.
OFS: OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.
ORS: ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

EXAMPLE

Use of NR built-in variables (Display Line Number)

student@fcait-ug-41:~/Desktop$ awk '{print NR,$0}' employee.txt

Use of NF built-in variables (Display Last Field)

student@fcait-ug-41:~/Desktop$ awk '{print $1,$NF}' employee.txt

Another use of NR built-in variables (Display Line From 3 to 6)

student@fcait-ug-41:~/Desktop$ awk 'NR==3, NR==6 {print NR,$0}' employee.txt