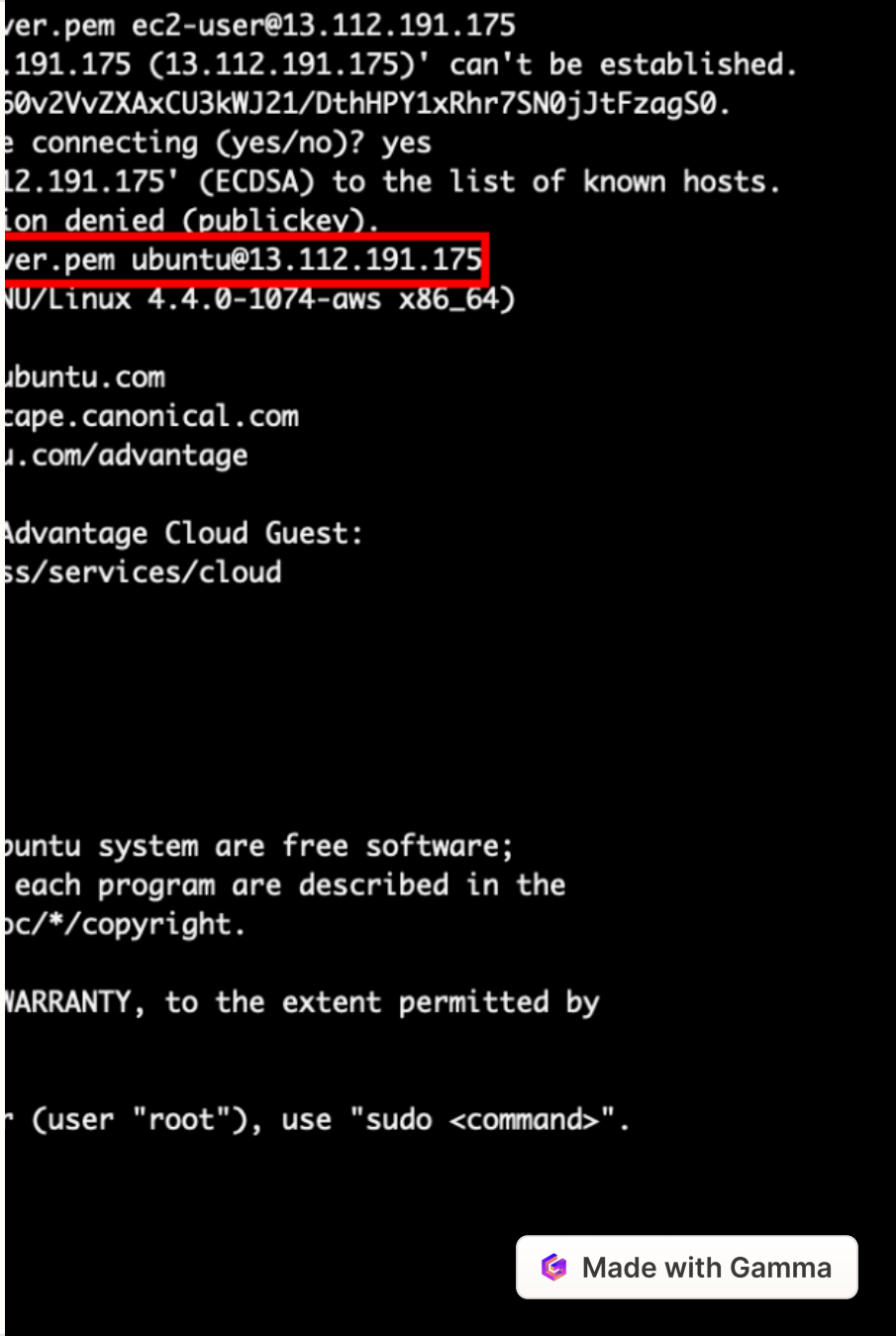
Text Processing Utilities

Text processing utilities in Linux are command-line tools for manipulating and analyzing text data. They include commands like `grep` (text search), `sed` (text stream editing), `awk` (text pattern processing), and others. These tools enable users to search, extract, transform, and analyze text files efficiently for various tasks.



1. grep: text-searching utility

```
vnr@vnrvjiet01:~$ cat f1
Hi am from aids
aids means artificial intelligence and data science
AIDS has high demand in future
Welcome!!

vnr@vnrvjiet01:~$ cat f2
aids class strength is 70
chatgpt is one of the application of ai

vnr@vnrvjiet01:~$ grep "aids" f1 f2
f1:Hi am from aids
f1:aids means artificial intelligence and data science
f2:aids class strength is 70
vnr@vnrvjiet01:~$ grep -i "aids" f1
Hi am from aids
aids means artificial intelligence and data science
AIDS has high demand in future
vnr@vnrvjiet01:~$ grep -v "Hi" f1
aids means artificial intelligence and data science
AIDS has high demand in future
Welcome!!

vnr@vnrvjiet01:~$ grep -v "aids" f1
AIDS has high demand in future
Welcome!!

vnr@vnrvjiet01:~$ grep -c "aids" f1
2
vnr@vnrvjiet01:~$
```

Description: The `grep` command in Linux is a text-searching utility used to search for patterns or text strings within files. It's a powerful tool for locating and displaying lines in a file that match a specified pattern.

Searching for a Word: grep "apple" fruits.txt

Case-Insensitive Search: grep -i "banana" fruits.txt

(The `-i` option makes the search case-insensitive, so it will match "banana" regardless of its case (e.g., "Banana" or "BANANA").

Searching Multiple Files: grep "keyword" file1.txt file2.txt

Inverting the Match: grep -v "error" log.txt

(The `-v` option inverts the match, displaying all lines that do not contain the specified pattern (e.g., lines without "error" in a log file).

Counting Matches: grep -c "apple" fruits.txt

(The `-c` option counts the number of matches rather than displaying the matching lines. It will output the count of occurrences of "apple" in "fruits.txt.")

Recursive Search in Directories: grep -r "keyword" /path/to/directory

(The `-r` (or `-R`) option searches recursively through all files in the specified directory and its subdirectories for the given pattern.)

2. sed: stream editor

```
vnr@vnrvjiet01:~$ ls
dir1  f1  f2
vnr@vnrvjiet01:~$ cat f1
Hi am from aids
aids means artificial intelligence and data science
AIDS has high demand in future
Welcome!!

vnr@vnrvjiet01:~$ cat f2
aids class strength is 70
chatgpt is one of the application of ai

vnr@vnrvjiet01:~$ sed 's/aids/AIDS/' f1
Hi am from AIDS
AIDS means artificial intelligence and data science
AIDS has high demand in future
Welcome!!

vnr@vnrvjiet01:~$ sed '/aids/d' f1
AIDS has high demand in future
Welcome!!

vnr@vnrvjiet01:~$ sed -n '/aids/p' f1
Hi am from aids
aids means artificial intelligence and data science
```

Description: The `sed` command, short for "stream editor," is a powerful text manipulation utility in Linux and Unix-like operating systems. `sed` operates on a line-by-line basis and allows you to perform various operations such as substitution, deletion, insertion, and more.

Syntax: sed [options] 'command' input_file

```
vnr@vnrvjiet01:~$ ls
dir1  f1  f2
vnr@vnrvjiet01:~$ cat f2
aids class strength is 70
chatgpt is one of the application of ai

vnr@vnrvjiet01:~$ sed -i 's/ai/Ai/' f2
vnr@vnrvjiet01:~$ cat f2
Aids class strength is 70
chatgpt is one of the application of Ai
```

Substitution (`s` command): sed 's/old_pattern/new_pattern/' input_file

Deletion (`d` command): sed '/pattern_to_delete/d' input_file

Printing (`p` **command):** sed -n '/pattern_to_print/p' input_file

The `p` command is used to print lines that match a pattern

Substituting and saving changes in-place (`-i` option): sed -i 's/old_pattern/new_pattern/' file.txt

Replacing all occurrences (`g` flag): sed 's/old_pattern/new_pattern/g' input_file

3. awk: pattern scanning& processing

Description: Awk is a scripting language used for manipulating data and generating reports. The awk command programming language requires no compiling and allows the user to use variables, numeric functions, string functions, and logical operators.

Syntax:

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

Print the lines which match the given pattern.

```
$ awk '/manager/ {print}' employee.txt
```

Splitting a Line Into Fields :

```
awk '{print $1,$4}' employee.txt
```

- **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.

```
vnr@vnrvjiet01:~$ ls
dir1  emp.txt  f1  f2
vnr@vnrvjiet01:~$ awk '{print}' emp.txt
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
vnr@vnrvjiet01:~$ awk '/manager/{print}' emp.txt
ajay manager account 45000
varun manager sales 50000
amit manager account 47000
vnr@vnrvjiet01:~$ awk '{print $1, $4}' emp.txt
ajay 45000
sunil 25000
varun 50000
amit 47000
vnr@vnrvjiet01:~$ awk '{print NR, $0}' emp.txt
1 ajay manager account 45000
2 sunil clerk account 25000
3 varun manager sales 50000
4 amit manager account 47000
vnr@vnrvjiet01:~$ awk '{print $1, NF}' emp.txt
ajay 4
sunil 4
varun 4
amit 4
vnr@vnrvjiet01:~$ awk '{print $1, $NF}' emp.txt
ajay 45000
sunil 25000
varun 50000
amit 47000
vnr@vnrvjiet01:~$ awk 'NR==2, NR==4 {print NR, $0}' emp.txt
2 sunil clerk account 25000
3 varun manager sales 50000
4 amit manager account 47000
vnr@vnrvjiet01:~$
```

```
vnr@vnrvjiet01:~$ awk '{print NR "- " $1 }' emp.txt
1- ajay
2- sunil
3- varun
4- amit
vnr@vnrvjiet01:~$ awk '{ if (length($0) > max) max = length($0) } END { print max }' emp.txt
26
```

- **NF:** NF command keeps a count of the number of fields within the current input record.

4. cut Command:

```
vnr@vnrvjiet01:~$ ls
dir1  emp.txt  f1  f2  state.txt
vnr@vnrvjiet01:~$ cat state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
vnr@vnrvjiet01:~$  cut -b 1,2,3 state.txt
And
Aru
Ass
Bih
Chh
vnr@vnrvjiet01:~$ cut -b 1-3,5-7 state.txt
Andra
Aruach
Assm
Bihr
Chhtti
```

Description: The `cut` command in Linux is used for cutting out sections from lines of files or data streams. It is often used to extract specific columns from text files, especially when working with delimited data like CSV files.

Syntax:

```
cut OPTION... [FILE]...
```

–complement: As the name suggests it complement the output. This option can be used in the combination with other options either with **-f** or with **-c**.

```
vnr@vnrvjiet01:~$ cut -c 2,5,7 state.txt
nr
rah
sm
ir
hti
vnr@vnrvjiet01:~$ cut -c 1-7 state.txt
Andhra
Arunach
Assam
Bihar
Chhatti
vnr@vnrvjiet01:~$ cut --complement -c 5 state.txt
Andha Pradesh
Arunchal Pradesh
Assa
Biha
Chhatisgarh
vnr@vnrvjiet01:~$ cut -f 1 state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
vnr@vnrvjiet01:~$ cut -d " " -f 1 state.txt
Andhra
Arunachal
Assam
Bihar
Chhattisgarh
```

1. -b(byte): To extract the specific bytes, you need to follow -b option with the list of byte numbers separated by comma.

2. -c (column): To cut by character use the -c option. This selects the characters given to the -c option.

3. -f (field): -c option is useful for fixed-length lines.

```
$cut -d "delimiter" -f (field number) file.txt
```

5. sort Command:

```
vnr@vnrvjiet01:~$ cat names.txt
abhishek
Rohith
ravi
pooja
Nithya
INS
vnr@vnrvjiet01:~$ sort names.txt
INS
Nithya
Rohith
abhishek
pooja
ravi
vnr@vnrvjiet01:~$ sort -r names.txt
ravi
pooja
abhishek
Rohith
Nithya
INS
```

```
vnr@vnrvjiet01:~$ cat num.txt
50
26
47
85
14
vnr@vnrvjiet01:~$ sort -n num.txt
14
26
47
50
85
vnr@vnrvjiet01:~$ sort -nr num.txt
85
50
47
26
14
```

Description: 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 the sort command can also be used to sort numerically.

Syntax:

```
$ sort filename.txt
```

```
vnr@vnrvjiet01:~$ cat > cars.txt
Audi
BMW
Cadillac
BMW
Dodge
vnr@vnrvjiet01:~$ sort -u cars.txt
Audi
BMW
Cadillac
Dodge
```

```
vnr@vnrvjiet01:~$ cat > employee.txt
manager  5000
clerk    4000
employee  6000
guard     3000
vnr@vnrvjiet01:~$ sort -k employee.txt
sort: invalid number at field start: in
vnr@vnrvjiet01:~$ sort -k 2 employee.tx
guard     3000
clerk    4000
manager  5000
employee  6000
```

```
vnr@vnrvjiet01:~$ cat cars.txt
Audi
BMW
Cadillac
BMW
Dodge
vnr@vnrvjiet01:~$ sort -u cars.txt | sort -c cars.txt
sort: cars.txt:4: disorder: BMW
vnr@vnrvjiet01:~$
```

Sort a File in Ascending Order: sort file.txt

Sort a File in Descending Order: sort -r file.txt

Sort Numerical Data: When sorting numerical data, use the `-n` option to sort numerically rather than lexicographically.

sort -n numbers.txt

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

-k Option: Unix provides the feature of sorting a table on the **basis of any column number by using -k option.**

-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.

-u option: To **sort and remove duplicates** pass the -u option to sort.

-M Option: To **sort by month** pass the -M option to sort.

6. tr: tr stands for translate.

```
vnr@vnrvjiet01:~$ echo "Hellooo" | tr -s 'o'
Hello
vnr@vnrvjiet01:~$ echo "Hello, World!" | tr -d ' ,!'
HelloWorld
vnr@vnrvjiet01:~$ echo "abc" | tr 'abc' 'xyz'
xyz
vnr@vnrvjiet01:~$
```

Description: 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. It can be used with UNIX pipes to support more complex translation.

Syntax:

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

```
vnr@vnrvjiet01:~$ cat > sm.txt
Welcome to
Linux utilities
vnr@vnrvjiet01:~$ cat sm.txt | tr [a-z] [A-Z]
WELCOME TO
LINUX UTILITIES
vnr@vnrvjiet01:~$ cat sm.txt | tr [:lower:] [:upper:]
WELCOME TO
LINUX UTILITIES
vnr@vnrvjiet01:~$ echo "Welcome To GeeksforGeeks" | tr [:space:] "\t"
Welcome To      GeeksforGeeks   vnr@vnrvjiet01:~$
vnr@vnrvjiet01:~$ echo "Welcome    To    GeeksforGeeks" | tr -s " "
Welcome To GeeksforGeeks
vnr@vnrvjiet01:~$ echo "my ID is 73535" | tr -d [:digit:]
my ID is
vnr@vnrvjiet01:~$ echo "my ID is 73535" | tr -cd [:digit:]
73535vnr@vnrvjiet01:~$
vnr@vnrvjiet01:~$ tr -cd [:digit:] <<< "my ID is 73535"
vnr@vnrvjiet01:~$
```

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

Conclusion

Text processing utilities in Linux, including commands like `grep`, `sed`, `awk`, and `cut`, offer essential tools for manipulating and analyzing text data efficiently. They empower users to search, extract, transform, and format text, making them indispensable for tasks like data parsing, log analysis, and report generation. With their command-line versatility and support for regular expressions, these utilities provide the means to efficiently process and extract valuable insights from text files. Whether handling delimited data, filtering lines, or performing complex substitutions, these tools streamline text-based operations and enhance productivity for both novice and experienced Linux users.