Filters in Linux

Subject Code:MCA13

Dr Manjunath M
Assistant Professor
Department of MCA
M SRamaiah Institute of Technology (MSRIT)
Bangalore - 560054

Mrs. Komala R
Assistant Professor
Department of MCA
M S Ramaiah Institute of Technology (MSRIT)
Bangalore - 560054

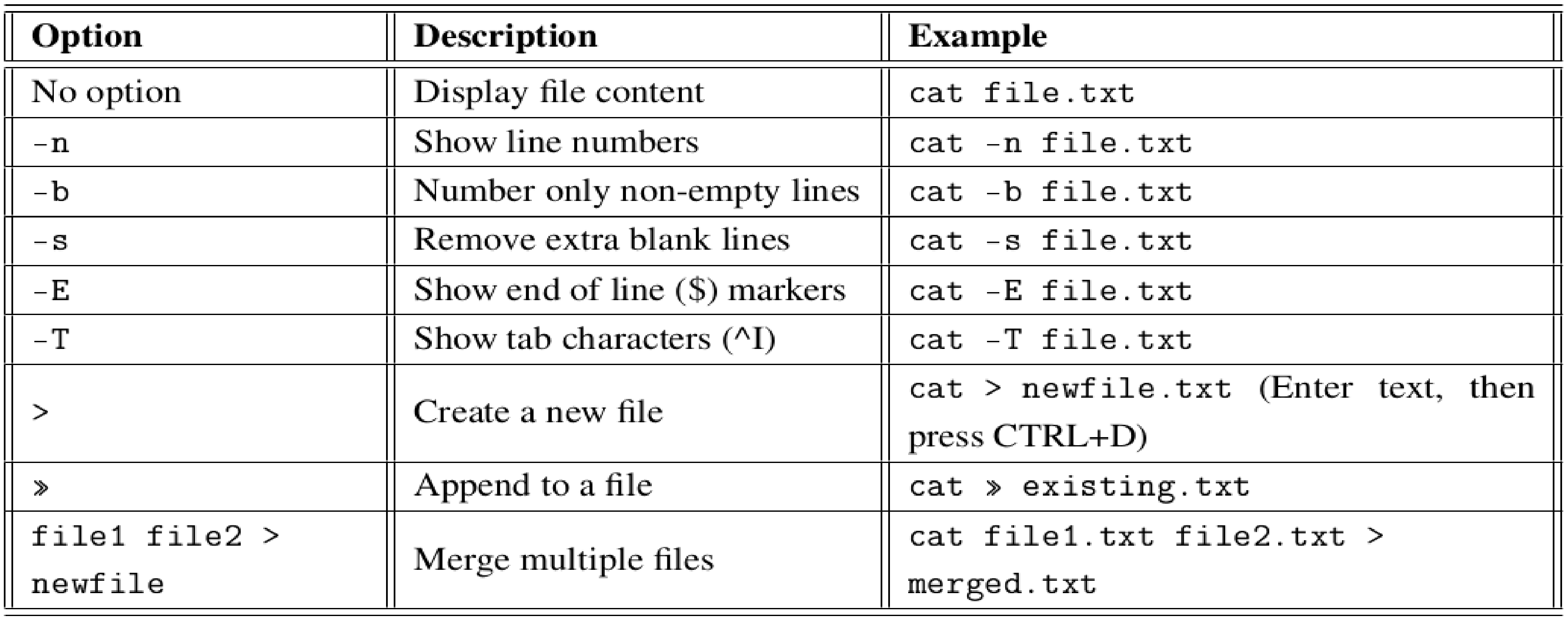
RAMAIAH Institute of Technology

Filters

- **Filters** are the set of commands that take **input from standard input stream i.e. stdin, perform some operations and write output to standard output stream i.e. stdout.**
- **Filters in Unix are commands that take input, process it, and produce output**,
- **Filters** typically used for **text processing**.
- They read data from **standard input (stdin), perform some transformation**, and **output the modified data to standard output (stdout)**.
- Common filter commands are
  - **cat**
  - **cut**
  - **head**
  - **tail**
  - **sort**
  - **Uniq**
  - **tr**

cat command

- The cat (concatenate) command is used to **view, combine, and create files**.
- **Syntax:**
  - **cat [options] filename**



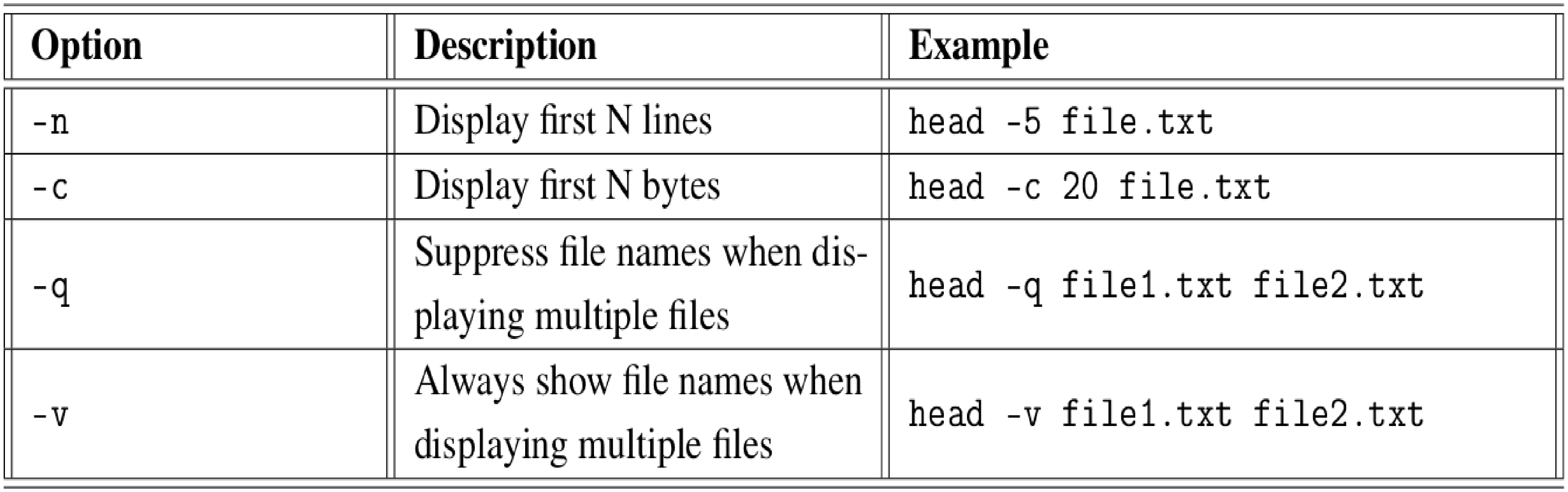
Option	Description	Example
No option	Display file content	`cat file.txt`
`-n`	Show line numbers	`cat -n file.txt`
`-b`	Number only non-empty lines	`cat -b file.txt`
`-s`	Remove extra blank lines	`cat -s file.txt`
`-E`	Show end of line ($) markers	`cat -E file.txt`
`-T`	Show tab characters (^I)	`cat -T file.txt`
`>`	Create a new file	`cat > newfile.txt` (Enter text, then press CTRL+D)
`»`	Append to a file	`cat » existing.txt`
`file1 file2 > newfile`	Merge multiple files	`cat file1.txt file2.txt > merged.txt`

Table 9: cat Command Options in Unix

RAMAIAH Institute of Technology

head command

- The **head command** is used to **display the first few lines or bytes of a file**.
- It is useful for quickly viewing the beginning of large text files.
- **Syntax:**
  - head [OPTION]... [FILE]...

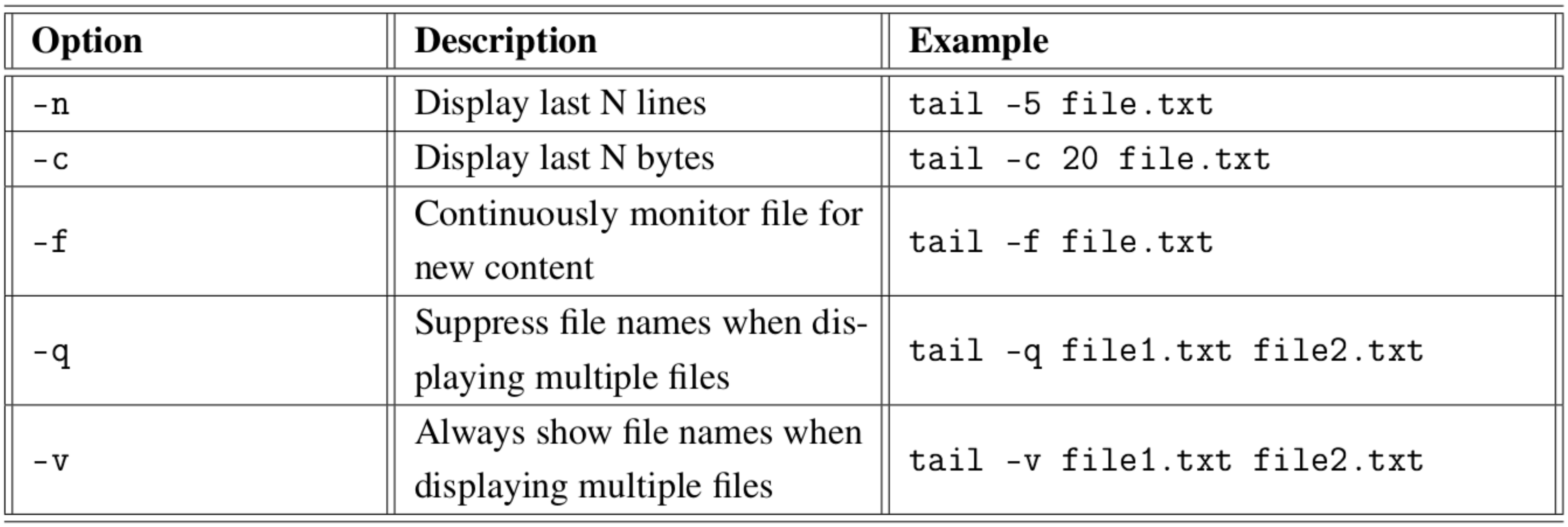


Option	**Description**	**Example**
`-n`	Display first N lines	`head -5 file.txt`
`-c`	Display first N bytes	`head -c 20 file.txt`
`-q`	Suppress file names when dis-playing multiple files	`head -q file1.txt file2.txt`
`-v`	Always show file names when displaying multiple files	`head -v file1.txt file2.txt`

Table 3: head Command Options in Unix

tail command

- The **tail command** is used to **display the last few lines or bytes of a file**.
- It is useful for viewing **logs, real-time updates, and recent data**.
- **Syntax:**
  - tail [OPTION]... [FILE]...

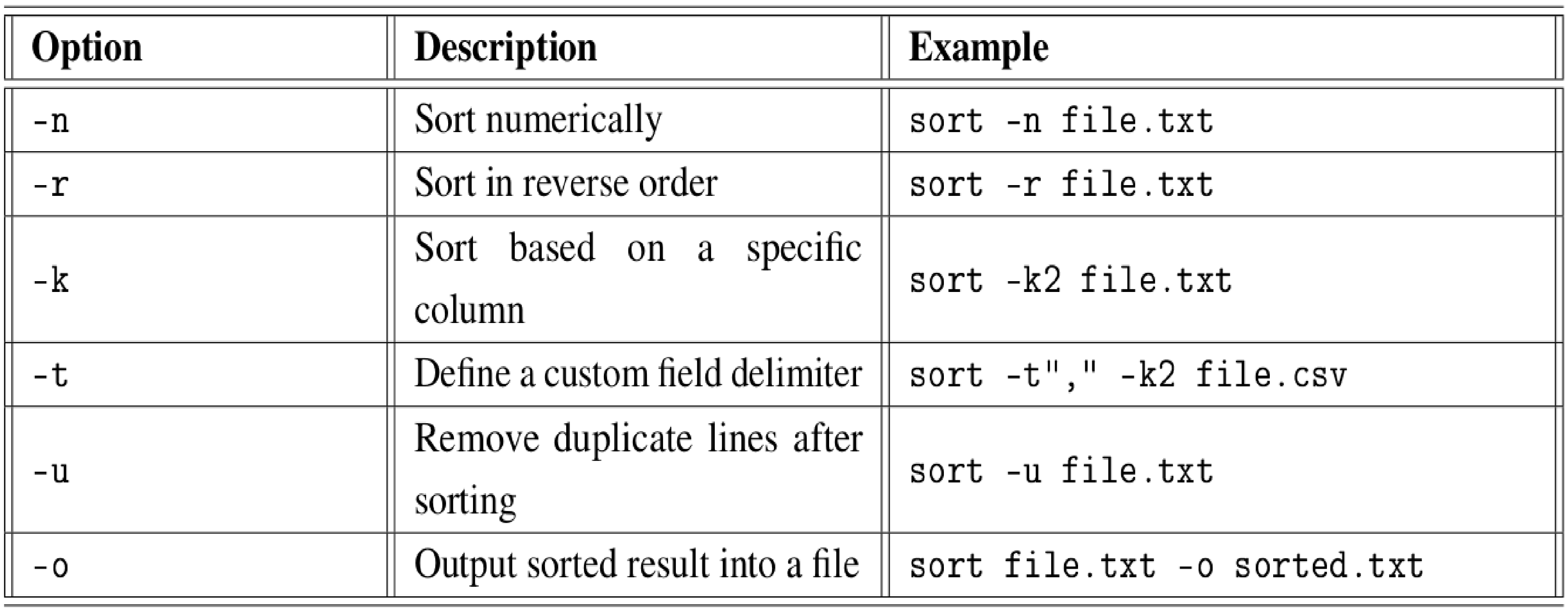


Option	Description	Example
`-n`	Display last N lines	`tail -5 file.txt`
`-c`	Display last N bytes	`tail -c 20 file.txt`
`-f`	Continuously monitor file for new content	`tail -f file.txt`
`-q`	Suppress file names when displaying multiple files	`tail -q file1.txt file2.txt`
`-v`	Always show file names when displaying multiple files	`tail -v file1.txt file2.txt`

Table 4: tail Command Options in Unix

sort command

- The sort command is used to arrange lines in text files in a specific order.
- **Syntax:**
  - tail [OPTION]... [FILE]...

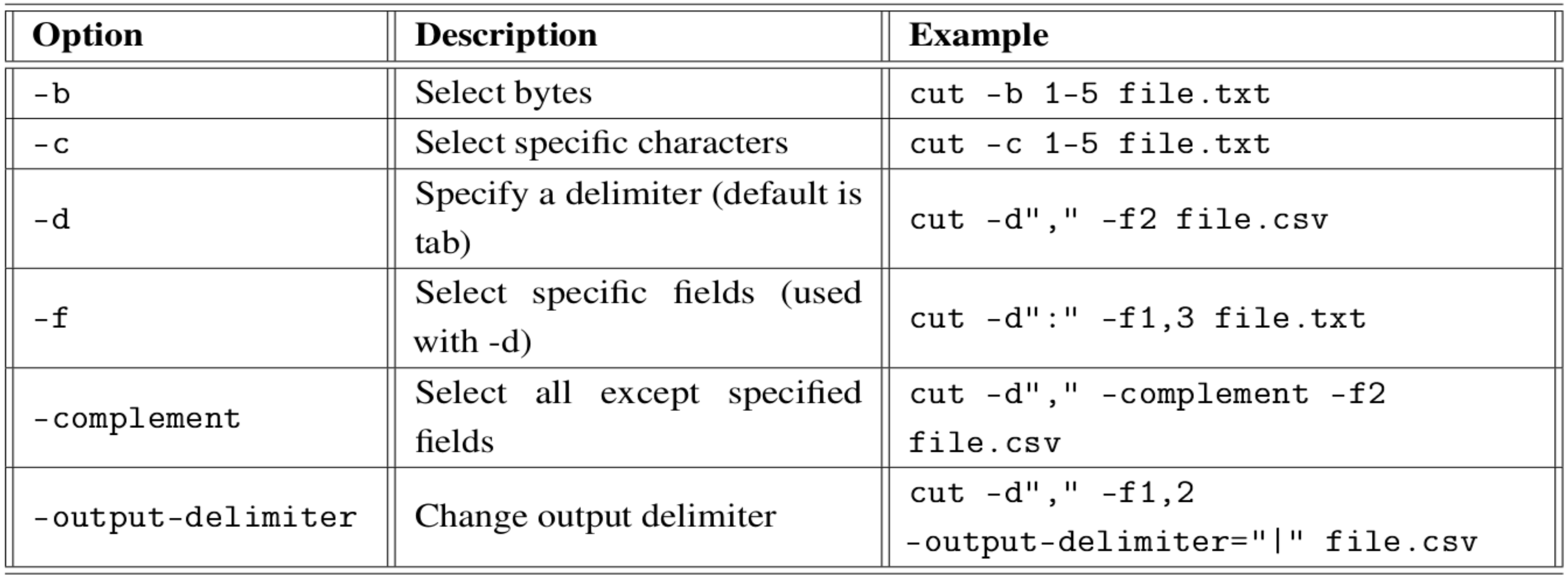


Option	Description	Example
-n	Sort numerically	`sort -n file.txt`
-r	Sort in reverse order	`sort -r file.txt`
-k	Sort based on a specific column	`sort -k2 file.txt`
-t	Define a custom field delimiter	`sort -t"," -k2 file.csv`
-u	Remove duplicate lines after sorting	`sort -u file.txt`
-o	Output sorted result into a file	`sort file.txt -o sorted.txt`

Table 5: sort Command Options in Unix

cut command

- The **cut command** in Unix is **used to extract specific sections of each line** from a file or standard input.
- It is commonly used for **text processing and works by selecting portions of data based on bytes, characters, or fields**.
- **Syntax:**
  - **cut [options] filename**

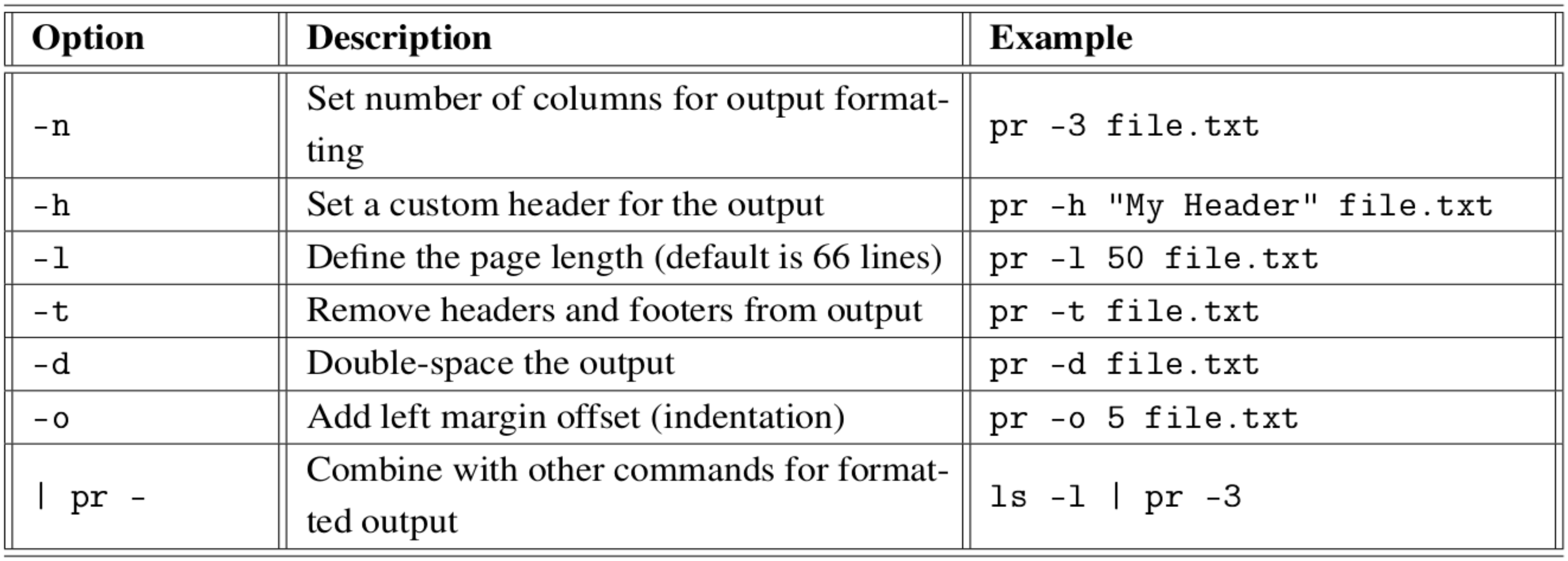


Option	Description	Example
`-b`	Select bytes	`cut -b 1-5 file.txt`
`-c`	Select specific characters	`cut -c 1-5 file.txt`
`-d`	Specify a delimiter (default is tab)	`cut -d"," -f2 file.csv`
`-f`	Select specific fields (used with -d)	`cut -d":" -f1,3 file.txt`
`-complement`	Select all except specified fields	`cut -d"," -complement -f2 file.csv`
`-output-delimiter`	Change output delimiter	`cut -d"," -f1,2 -output-delimiter="|" file.csv`

Table 2: cut Command Options in Unix

pr command

- The **pr** command in Linux is used to **format text files for printing**.
- It **adds headers, footers, page breaks, columns, and more** to make output look structured when printed.
- **Syntax:**
  - **pr [options] [file]**

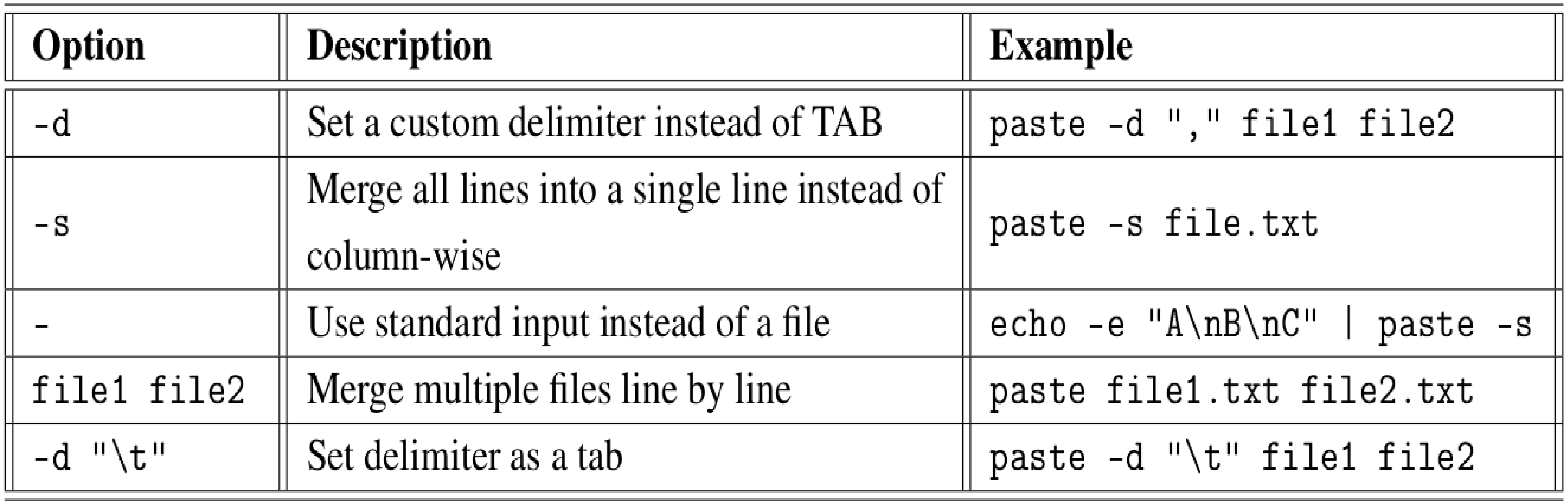


Option	Description	Example
`-n`	Set number of columns for output formatting	`pr -3 file.txt`
`-h`	Set a custom header for the output	`pr -h "My Header" file.txt`
`-l`	Define the page length (default is 66 lines)	`pr -l 50 file.txt`
`-t`	Remove headers and footers from output	`pr -t file.txt`
`-d`	Double-space the output	`pr -d file.txt`
`-o`	Add left margin offset (indentation)	`pr -o 5 file.txt`
`| pr -`	Combine with other commands for formatted output	`ls -l | pr -3`

Table 6: pr Command Options in Linux

paste command

- The **paste** command in Linux is used to **merge lines of files horizontally (side by side)** by joining them column-wise.
- **Syntax:**
  - **paste [options] file1 file2 ...**

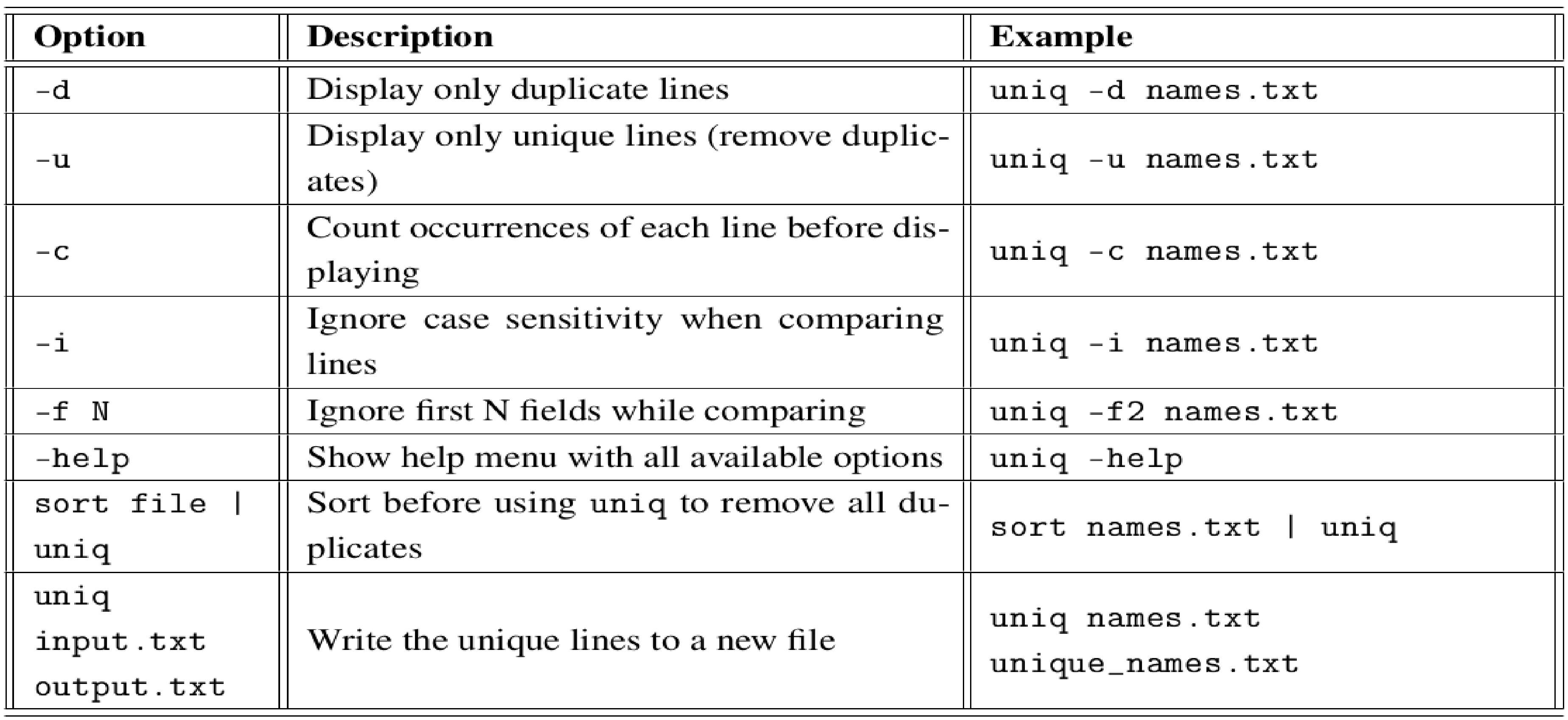


Option	Description	Example
`-d`	Set a custom delimiter instead of TAB	`paste -d "," file1 file2`
`-s`	Merge all lines into a single line instead of column-wise	`paste -s file.txt`
`-`	Use standard input instead of a file	`echo -e "A\nB\nC" | paste -s`
`file1 file2`	Merge multiple files line by line	`paste file1.txt file2.txt`
`-d "\t"`	Set delimiter as a tab	`paste -d "\t" file1 file2`

Table 7: paste Command Options in Linux

uniq command

- The **uniq** command in Linux is used to **filter out adjacent duplicate lines from a sorted file or input**.
- It helps in detecting and removing consecutive duplicate entries while keeping the first occurrence.
- **Syntax: uniq [options] file1**



Option	Description	Example
`-d`	Display only duplicate lines	`uniq -d names.txt`
`-u`	Display only unique lines (remove duplicates)	`uniq -u names.txt`
`-c`	Count occurrences of each line before displaying	`uniq -c names.txt`
`-i`	Ignore case sensitivity when comparing lines	`uniq -i names.txt`
`-f N`	Ignore first N fields while comparing	`uniq -f2 names.txt`
`-help`	Show help menu with all available options	`uniq -help`
`sort file | uniq`	Sort before using uniq to remove all duplicates	`sort names.txt | uniq`
`uniq input.txt output.txt`	Write the unique lines to a new file	`uniq names.txt unique_names.txt`

Table 8: uniq Command Options in Linux

RAMAIAH Institute of Technology

uniq command

Option	Description	Example
`-d`	Display only duplicate lines	`uniq -d names.txt`
`-u`	Display only unique lines (remove duplicates)	`uniq -u names.txt`
`-c`	Count occurrences of each line before displaying	`uniq -c names.txt`
`-i`	Ignore case sensitivity when comparing lines	`uniq -i names.txt`
`-f N`	Ignore first N fields while comparing	`uniq -f2 names.txt`
`-help`	Show help menu with all available options	`uniq -help`
`sort file | uniq`	Sort before using uniq to remove all duplicates	`sort names.txt | uniq`
`uniq input.txt output.txt`	Write the unique lines to a new file	`uniq names.txt unique_names.txt`

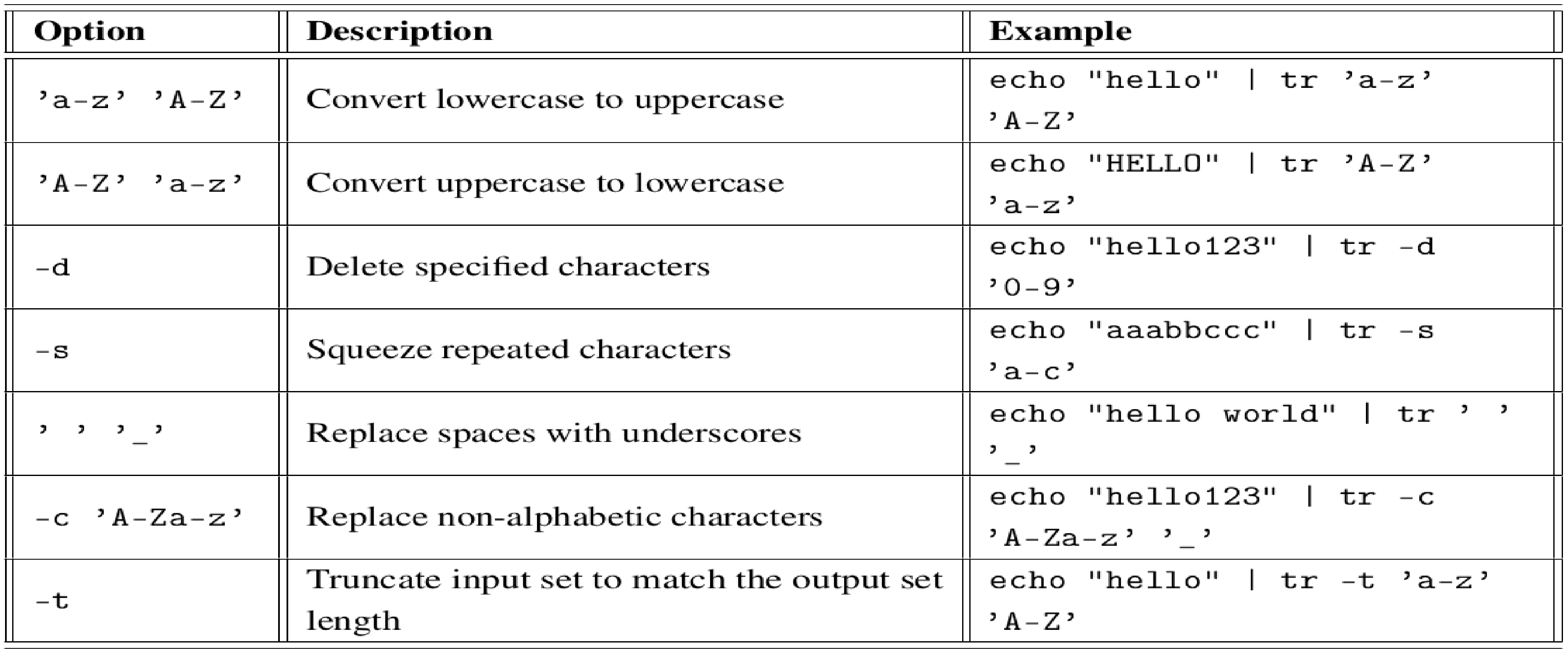
Table 8: uniq Command Options in Linux

Command	What It Does
`uniq -f1 file.txt`	Ignores the first field and removes duplicates based on the rest.
`uniq -f2 file.txt`	Ignores the first 2 fields and checks duplicates from field 3 onward.
`uniq file.txt`	Removes exact duplicate lines (does not ignore fields).

Table 9: uniq Command Options in Linux

tr command

- The **tr (translate)** command in Linux is used for **text transformation by replacing, deleting, or compressing characters** from standard input (stdin).
- **Syntax:**
  - **tr [OPTION] SET1 [SET2]**



Option	Description	Example
’a-z’ ’A-Z’	Convert lowercase to uppercase	echo "hello" | tr ’a-z’ ’A-Z’
’A-Z’ ’a-z’	Convert uppercase to lowercase	echo "HELLO" | tr ’A-Z’ ’a-z’
-d	Delete specified characters	echo "hello123" | tr -d ’0-9’
-s	Squeeze repeated characters	echo "aaabbccc" | tr -s ’a-c’
’ ’ ’_’	Replace spaces with underscores	echo "hello world" | tr ’ ’ ’_’
-c ’A-Za-z’	Replace non-alphabetic characters	echo "hello123" | tr -c ’A-Za-z’ ’_’
-t	Truncate input set to match the output set length	echo "hello" | tr -t ’a-z’ ’A-Z’

Table 10: tr Command Options in Linux

RAMAIAH Institute of Technology

grep command

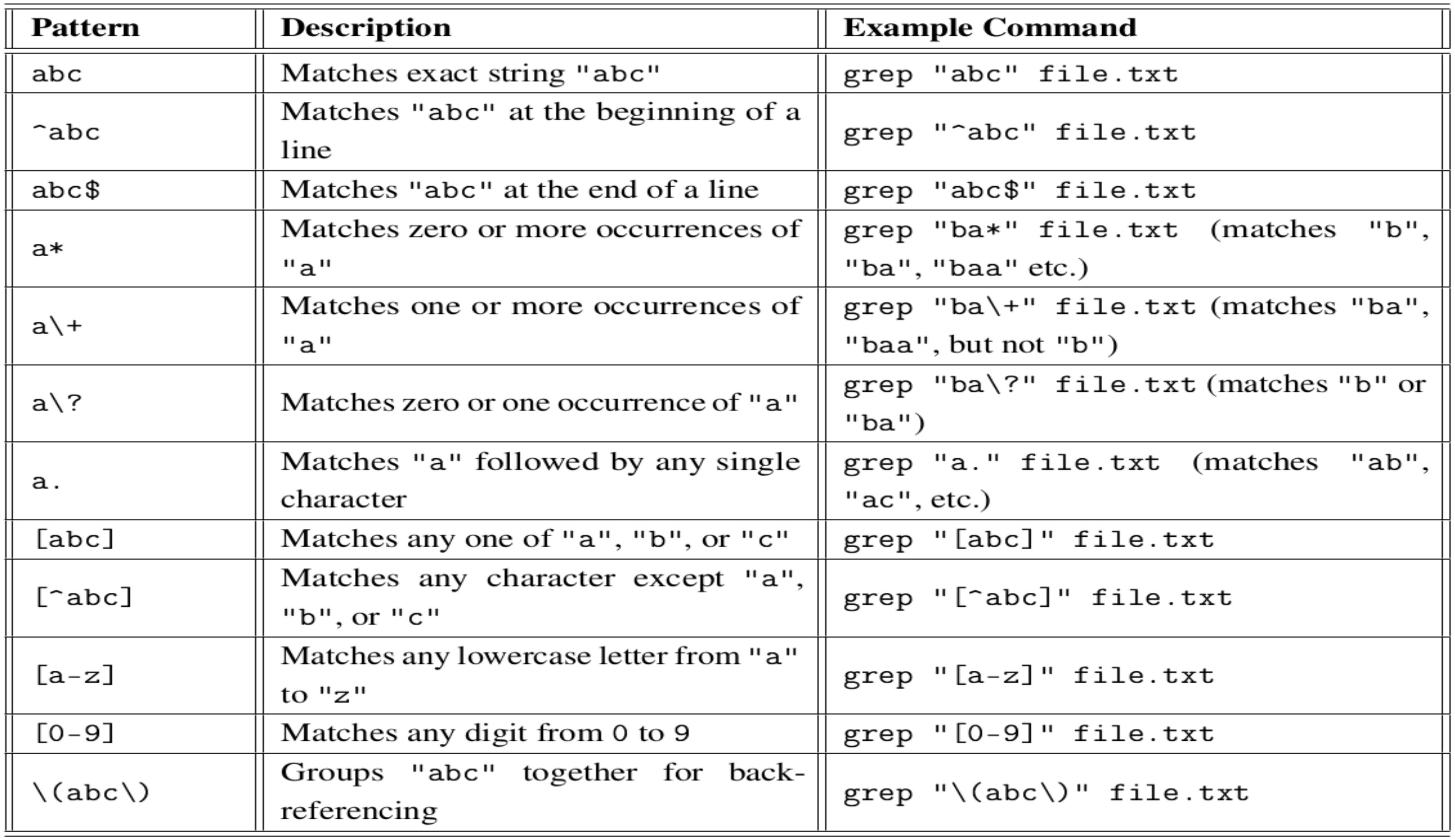
- The grep stands for Global Regular Expression Print
- The grep command in Linux is used to search for specific text or patterns in files or input streams.
- Syntax:
  - grep [OPTIONS] PATTERN [FILE]
- Example:
  - grep -c "hello" file.txt

grep command

Option	Description	Example
`-c`	Count the number of lines that match the pattern	`grep -c "error" logfile.txt`
`-h`	Suppress filename output when searching multiple files	`grep -h "error" *.log`
`-l`	Display only filenames of files that contain the search pattern	`grep -l "error" *.log`
`-n`	Display line numbers along with matching lines	`grep -n "error" logfile.txt`
`-v`	Invert match to show lines that do not contain the pattern	`grep -v "success" logfile.txt`
`-o`	Display only the matched text instead of the full line	`grep -o "error" logfile.txt`
`-e`	Use extended regular expressions for complex patterns	`grep -e "error|fail" logfile.txt`
`^`	Match lines that start with a specific pattern	`grep "^Start" logfile.txt`
`$`	Match lines that end with a specific pattern	`grep "End$" logfile.txt`
`-i`	Perform case-insensitive search	`grep -i "error" logfile.txt`

Table 11: grep Command Options in Linux

BRE command



Pattern	Description	Example Command
abc	Matches exact string "abc"	grep "abc" file.txt
^abc	Matches "abc" at the beginning of a line	grep "^abc" file.txt
abc$	Matches "abc" at the end of a line	grep "abc$" file.txt
a*	Matches zero or more occurrences of "a"	grep "ba*" file.txt (matches "b", "ba", "baa" etc.)
a\+	Matches one or more occurrences of "a"	grep "ba\+" file.txt (matches "ba", "baa", but not "b")
a\?	Matches zero or one occurrence of "a"	grep "ba\?" file.txt (matches "b" or "ba")
a.	Matches "a" followed by any single character	grep "a." file.txt (matches "ab", "ac", etc.)
[abc]	Matches any one of "a", "b", or "c"	grep "[abc]" file.txt
[^abc]	Matches any character except "a", "b", or "c"	grep "[^abc]" file.txt
[a-z]	Matches any lowercase letter from "a" to "z"	grep "[a-z]" file.txt
[0-9]	Matches any digit from 0 to 9	grep "[0-9]" file.txt
\(abc\)	Groups "abc" together for back-referencing	grep "\(abc\)" file.txt

Table 12: Basic Regular Expressions (BRE) for ‘grep‘ in Linux

Extended Regular Expression (ERE)

Extended Regular Expression (ERE)

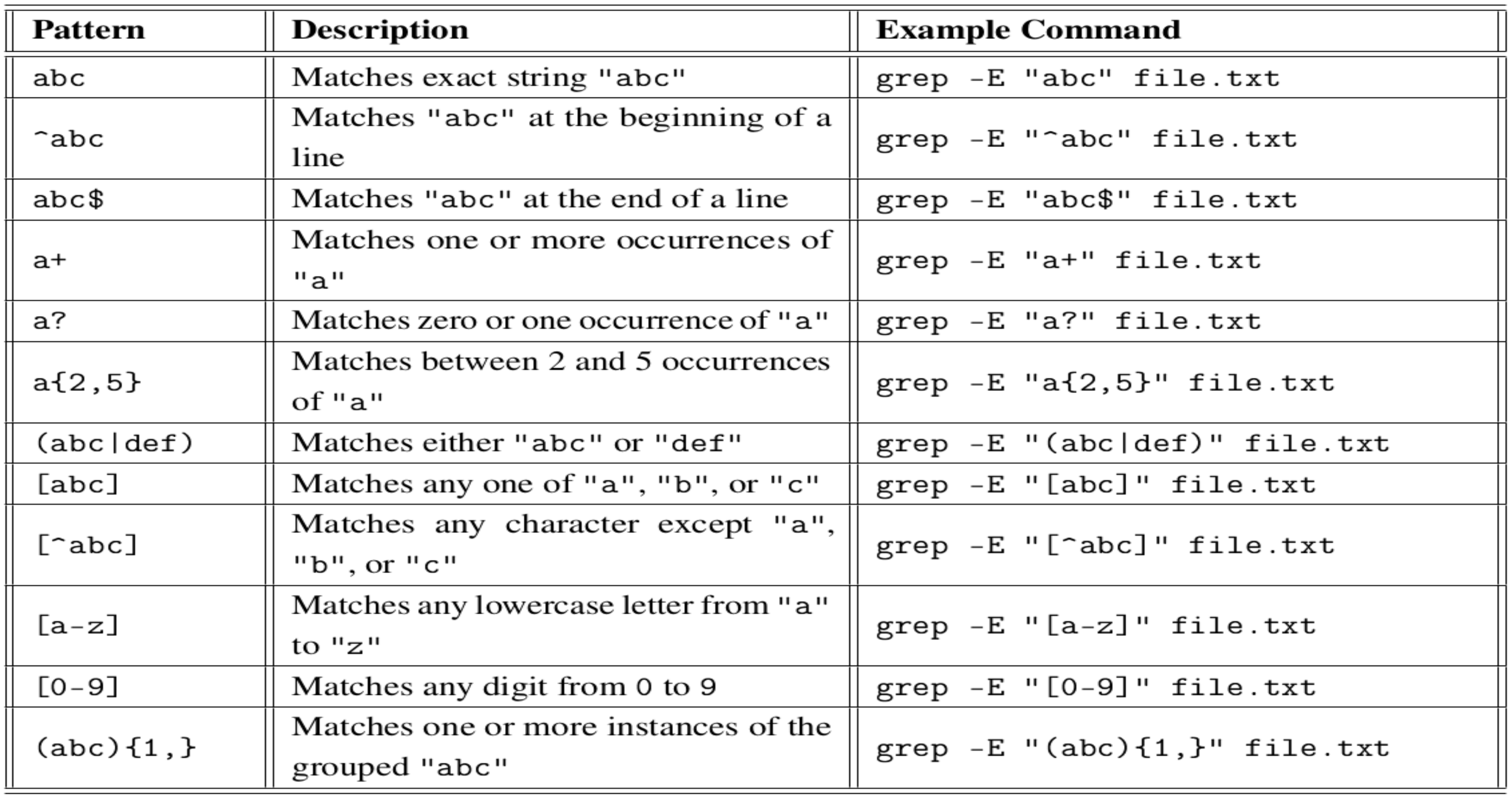
- **Extended Regular Expressions (ERE) introduce several enhancements over Basic Regular Expressions (BRE) that facilitate more powerful and flexible pattern** matching without the verbose syntax often required in BRE.

- The **primary enhancements of ERE over BRE involve adding syntactic sugar (e.g., +, ?, and |)** that reduces the need for escaping and simplifies expressions.

- ERE also typically **processes patterns slightly faster due to their optimized handling** of these expressions.

- These features make ERE particularly useful for complex pattern matching tasks in scripts and command-line operations where readability and efficiency are crucial.

RE command



Pattern	Description	Example Command
abc	Matches exact string "abc"	grep -E "abc" file.txt
^abc	Matches "abc" at the beginning of a line	grep -E "^abc" file.txt
abc$	Matches "abc" at the end of a line	grep -E "abc$" file.txt
a+	Matches one or more occurrences of "a"	grep -E "a+" file.txt
a?	Matches zero or one occurrence of "a"	grep -E "a?" file.txt
a{2,5}	Matches between 2 and 5 occurrences of "a"	grep -E "a{2,5}" file.txt
(abc|def)	Matches either "abc" or "def"	grep -E "(abc|def)" file.txt
[abc]	Matches any one of "a", "b", or "c"	grep -E "[abc]" file.txt
[^abc]	Matches any character except "a", "b", or "c"	grep -E "[^abc]" file.txt
[a-z]	Matches any lowercase letter from "a" to "z"	grep -E "[a-z]" file.txt
[0-9]	Matches any digit from 0 to 9	grep -E "[0-9]" file.txt
(abc){1,}	Matches one or more instances of the grouped "abc"	grep -E "(abc){1,}" file.txt

Table 20: Extended Regular Expressions (ERE) for ‘grep -E‘ in Linux

RAMAIAH Institute of Technology

End of the Presentation