School of Computer Science Engineering and Technology
Bennett University (The Times Group)

Program- B. Tech-3rd Semester
Course Code- CSET213
Year- 2025
Date- 02/09/2025

Type- Sp. Core-I
Course Name-Linux and Shell Programming
Semester- Odd
Batch- Cyber Security (B1-14)

Lab Assignment 8

Exp No	Name	CO1	CO 2	CO3	CO4
8	Shell programming environments- filters and pipe	✓ -	✓	-	-

Objective: To learn the usage of pipes and filters by executing some interesting examples.

Outcomes: After hands-on, you will be able to differentiate between redirection and pipe. Moreover, you can apply the concept of pipe when the output of one command can be the input of another command, and we need to do it without storing the output of the former command in a temporary file.

Hands-on Learning on pipes and filters (60 minutes)

Redirecting Terminal I/O: Whenever a shell command runs, three standard file streams are opened like **"standard in (stdin)" for input; "standard out (stdout)"; and "standard error(stderr)" for output.** The shell feature known as redirection is used to assign files other than the user terminal to a standard stream. The formats for redirection are:

<command> < *<file-name>*	[*<command>* gets input from *<file-name>* rather than the terminal]
<command> > *<file-name>*	[*<command>* sends output to *<file-name>* rather than the terminal]
<command> >> *<file-name>*	[*<command>* appends output to *<file-name>*]

Examples:

- sort > mysortedfile
- sort file1 file2 > mysortedfile
- sort 0<filein 1>mysortedfile 2>errlist
- sort filein > mysortedfile 2> errlist
- sort filein >mysortedfile 2>&1
- (date; cal) > myfile

1. Why Pipes?

Redirection provides a shell command with an alternative to the user terminal for standard

input, standard output, or standard error. Redirection can't be used to take standard output and turn it directly into standard input for another command without going through an intermediate file.

Let us consider the example shown below. The objective of doing "sort < ls -l" would need to be accomplished by

```
ls -l > temp
sort < temp
```

Example1:

By the use of pipe (|), the shell provides a means of taking standard output from one command and making it standard input for another without using any temporary file.

The notation of pipe is:

<command-1> | <command-2> **(1)**

Equation-1 is used to denote that standard output from <command-1> is to be piped to <command-2> as standard input.

For the example 1, using pipe, you can simply execute:

ls -l | sort

2. Filters

A filter is a command that takes data from a file and performs some simple transformation on it, the result of which is sent on to some other file.

Examples of filter:

sort - sort files

grep (and its derivatives) - search for keyword information in the file

head - output lines from the front end of the file

tail - output lines from the tail end of the file

wc - count words, lines, and/or characters in the file

2.1. grep (global regular expression - print) is a shell command that matches patterns as represented by limited regular expressions against the input character stream.

Example: grep -n 'Linux' BennettCourseFile.docx

gives the line numbers and prints the lines containing the specific symbol "Linux".

2.1.1 Metacharacters used with grep

Metacharacters are used for the mechanisms employed by regular expressions to represent complex patterns, e.g.

^	for the beginning of a line	'^t' (lines beginning with "t")

$	for the end of a line	't$' (lines ending with "t")
.	matches any single character	'^.t' (lines with 1st character anything and 2nd character "t")
*	goes with the preceding character to represent 0 or more repetitions of the character	
+	is like *, but is for 1 or more repetitions	
\	turns off any special meaning for the character that follows it	

2.1.2 Construction rules for regular expressions provide means for using simpler regular expressions to define more complex regular expressions. Example:

> [...] match is a regular expression: match is to any character listed; allows ranges such as a-x.
>
> [^...] not match is a regular expression: match is to any character not listed; also allows ranges.

2.2 head

Example: **head -20 myFile.txt** //lists the first 20 lines of myFile.txt
Note: If no number is specified, the default is 10.

2.3 tail

Example: **tail -20 myFile.txt** //lists the last 20 lines of myFile.txt
Note: If no number is specified, the default is 10.

2.4 wc

Example: **wc -lwc myFile.txt** //counts lines, words, and characters in the file.

Note: If any one of l, w, or c is omitted, that count is not provided.

Scripting Problems for Assessment (60 Minutes)

1. Create a file name “moviefileName.txt” having the names of the Science Fiction movies.
    i. Consider some redundant data is present in the file. So, remove redundancy and store that in moviefileName_RemovedRedundancy.txt
    ii. The data in moviefileName_RemovedRedundancy.txt is unordered. So, order them in ascending order and preserve output in moviefileNameSortedInAscending.txt.
    iii. Filter out the movie names that starts with a and ends with any vowel.
    iv. Display the movie names having alphabets ‘d’ and ‘e’ combined in the word. Example “Spi**de**rhead” (15 Minutes) **(Even Batch)**
2. Create a file with name “bigdata.csv”. Keep 1000 numeric values in it. You need to show and preserve in stdout and file respectively:

- First 100 values of 1000 data
- Last 100 values of 1000 data
- Filter out those numbers whose first and last digit is 1
- Filter out those numbers whose all digits are 9

Note: You have to create a file using Linux command. You need to put 10000 random numeric data shuffled using the code that can be in any language. (20 Minutes) **(Odd Batch)**

3. Write a shell script to print file names in directory showing date of creation & serial no. of file. (10 Minutes) **(Even Batch)**

4. Display all the link files present in the current directory. (10 Minutes)
Note: You have to create 100 subdirectories and create links for each of them and store it in different directory (let us suppose Desktop). So, in total you must have 100 links in the Desktop directory. **(Odd Batch)**

Submission Instructions:

1. Submission requires the screen shots of all the incurred steps to execute a shell script or a video showing the whole process.
2. All these files are in single zip folder.
3. Use the naming convention: Prog_CourseCode_RollNo_LabNo.docx (Example: BCA3rdSem_CBCA221_ E21BCA002_Lab1.1)
4. Submission is through LMS only