Lab No. 04

Text Processing and Pipelining

Objective:

This lab will introduce the Text Processing Tools available in bash shell along with some basic system configuration commands.

Activity Outcomes:

On completion of this lab students will be able to:

- Use some of the most useful text-processing utilities
- Write complex commands using piplines

Instructor Notes

As pre-lab activity, read Chapter 16 to 20 from the book "The Linux Command Line", William E. Shotts, Jr.

1) Useful Concepts

Text Processing Utilities

In first part of this lab, we will discuss some basic and useful utilities available in bash shell

Viewing File Contents with less

The less command is a program to view text files. Throughout our Linux system, there are many files that contain human-readable text. The less program provides a convenient way to examine them. Why would we want to examine text files? Because many of the files that contain system settings (called configuration files) are storedin this format and being able to read them gives us insight about how the system works. In addition, many of the actual programs that the system uses (called scripts) are stored in this format.
The less command is used like this:

```
$less <filename>
```

Once started, the less program allows you to scroll forward and backward through a text file. For example, to examine the file that defines all the system's user accounts, enterthe following command:

```
$less /etc/passwd
```

Once the less program starts, we can view the contents of the file. If the file is longer than one page, we can scroll up and down. To exit less, press the "q" key.

The table below lists the most common keyboard commands used by less.

Command	Action
Page Up or b	Scroll back one page
Page Down or space	Scroll forward one page
Up Arrow	Scroll up one line
Down Arrow	Scroll down one line
G	Move to the end of the text file
1G or g	Move to the beginning of the text file
/*characters*	Search forward to the next occurrence of *characters*
n	Search for the next occurrence of the previous search
h	Display help screen
q	Quit `less`

wc – Print Line, Word, and Byte Counts

The wc (word count) command is used to display the number of lines, words, and bytescontained in files. For example:

```
$wc Myfile.txt
```

In this case it prints out three numbers: lines, words, and bytes contained in Myfile.txt.

Pipelines

The ability of commands to read data from standard input and send to standard output is utilized by a shell feature called pipelines. Using the pipe operator "|" (vertical bar), the standard output of one command can be piped into the standard input of another: For example, we can use less to display, page-by-page, the output of any command that sends its results to standard output:

```
$ls -l /usr/bin | less
```

Filters

Pipelines are often used to perform complex operations on data. It is possible to put severalcommands together into a pipeline. Frequently, the commands used this way are referredto as filters. Filters take input, change it somehow and then output it. The first one we will try is **sort**.

Sort

Imagine we wanted to make a combined list of all of the executable programs in /bin and /usr/bin, put them in sorted order and view it:

```
$ls  /bin /usr/bin | sort | less
```

Since we specified two directories (/bin and /usr/bin), the output of ls would have consisted of two sorted lists, one for each directory. By including sort in our pipeline,we changed the data

to produce a single, sorted list.

grep – Print Lines Matching A Pattern

grep is a powerful program used to find text patterns within files. It's used like this:

```
grep pattern [file...]
```

When grep encounters a “pattern” in the file, it prints out the lines containing it. The patterns that grep can match can be very complex, but for now we will concentrate onsimple text matches. For example, find all the files in our list of programs that had the word “zip” embedded in the name:

```
$ ls /bin /usr/bin | sort | grep zip
```

There are a couple of handy options for grep: “-i” which causes grep to ignore case when performing the search (normally searches are case sensitive) and “-v” which tells grep to only print lines that do not match the pattern.

head / tail – Print First / Last Part of Files

Sometimes you don't want all the output from a command. You may only want the first fewlines or the last few lines. The head command prints the first ten lines of a file and the tail command prints the last ten lines. By default, both commands print ten lines of text, but this can be adjusted with the “-n” option:

```
$head -n 5 Myfile.txt
$tail -n 5 Myfile.txt
```

These can be used in pipelines as well:

```
$ ls /usr/bin | tail -n 5
```

Examining and Monitoring A Network

The ping command sends a special network packet called an ICMP ECHO_REQUEST to a specified host. Most network devices receiving this packet will reply to it, allowing the network connection to be verified.
For example

```
$ping localhost
```

After it is interrupted by pressing Ctrl-c, ping prints performance statistics. A properly performing network will exhibit zero percent packet loss. A successful “ping” will indicate that the elements of the network (its interface cards, cabling, routing, and gateways) are ingenerally good working order.

Set Date and Time

Display Current Date and Time: Just type the date command:

```
$ date
```

Sample outputs:

Mon Jan 21 01:31:40 IST 2019

Display the Hardware Clock (RTC): Type the following hwclock command to read the Hardware Clock and display the time onscreen:

```
# hwclock -r OR
# hwclock --show
$ sudo hwclock --show --verbose
```

OR show it in Coordinated Universal time (UTC):

```
# hwclock --show --utc
```

Sample outputs:

2019-01-21 01:30:50.608410+05:30

Set Date Command: Use the following syntax to set new data and time:

```
date --set "STRING"
```

For example, set new data to 2 Oct 2006 18:00:00, type the following command as rootuser:

```
# date -s "2 OCT 2006 18:00:00" OR
# date --set "2 OCT 2006 18:00:00"
```

You can also simplify format using following syntax:# date +%Y%m%d -s "20081128"

Set Time Examples To set time use the following syntax:

```
# date +%T -s "10:13:13"
```

Where,

1: Hour (hh)

2: Minute (mm)

3: Second (ss)

Use %p locale's equivalent of either AM or PM, enter:

```
# date +%T%p -s "6:10:30AM" # date +%T%p -s "12:10:30PM"
```

Synchronizing Hardware and System Clocks: Use the following syntax:

```
# hwclock --systohc OR
# hwclock -w
```

timedatectl: Display the current date and timeType the following command:

```
$ timedatectl
```

To change the current date, type the following command as root user:

```
 # timedatectl set-time YYYY-MM-DD
```

OR

$ sudo timedatectl set-time YYYY-MM-DD

For example set the current date to 2015-12-01 (1st, Dec, 2015):

```
# timedatectl set-time '2015-12-01'
# timedatectl
```

Sample outputs:

Local time: Tue 2015-12-01 00:00:03 EST

Universal time: Tue 2015-12-01 05:00:03 UTC

RTC time: Tue 2015-12-01 05:00:03

Time zone: America/New_York (EST, -0500)NTP enabled: no

NTP synchronized: noRTC in local TZ: no

DST active: no

Last DST change: DST ended at

Sun 2015-11-01 01:59:59 EDT

Sun 2015-11-01 01:00:00 EST

Next DST change: DST begins (the clock jumps one hour forward) atSun 2016-03-13 01:59:59 EST

Sun 2016-03-13 03:00:00 EDT

To change both the date and time, use the following syntax:

```
# timedatectl set-time YYYY-MM-DD HH:MM:SS

          Where,
```

HH : An hour.

MM : A minute.

SS : A second, all typed in two-digit form.

YYYY: A four-digit year.

MM : A two-digit month.

DD: A two-digit day of the month.

For example, set the date ’23rd Nov 2015′ and time to ‘8:10:40 am’, enter:

```
# timedatectl set-time '2015-11-23 08:10:40'

# date
```

Set the current time: the syntax is

```
# timedatectl set-time HH:MM:SS # timedatectl set-time
'10:42:43'
# date
```

Sample outputs:

Mon Nov 23 08:10:41 EST 2015

How do I set the time zone using timedatectl command?To see list all available time zones, enter:

```
$ timedatectl list-timezones
$ timedatectl list-timezones | more
$ timedatectl list-timezones | grep -i asia
$ timedatectl list-timezones | grep America/New To set the time zone
to 'Asia/Kolkata',
```

Enter:

```
# timedatectl set-timezone 'Asia/Kolkata'
```

Verify it:

```
# timedatectl
```

Local time: Mon 2015-11-23 08:17:04 IST

Universal time: Mon 2015-11-23 02:47:04 UTC

RTC time: Mon 2015-11-23 13:16:09

Time zone: Asia/Kolkata (IST, +0530)NTP enabled: no
NTP synchronized: noRTC in local TZ: no
DST active: n/aLab Activities

2) Solved Lab Activities

Sr.No	*Allocated Time*	*Level of Complexity*	*CLO Mapping*
1	*15*	*Low*	*CLO-5*
2	*15*	*Medium*	*CLO-5*
3	*5*	*Low*	*CLO-5*

Activity 1:

This activity is related to file permission. Perform the following tasks

- ***Create a file "testfile.txt" in /test directory***
- ***View and read the file using Less command***
- ***Use wc command to print lines, words and bytes***
- ***Find any text pattern in the file using grep***
- ***Print the first 3 lines of the file using head***
- ***Print the last 3 lines of the file using tail***

Solution:

- `touch /test/testfile.txt`
- `less testfile.txt`
- `wc testfile.txt`
- `grep testfile.txt`
- `head -n 3 testfile.txt`
- `tail -n 3 testfile.txt`

Activity 2:

Perform the following tasks

- ***Find all the files with extension txt in the /test directory***
- ***Find the first line of the list of files in the /test directory***
- ***Find the last line of the list of files in the /test directory***
- ***Produce and view a single sorted list of files by combining two directories: /Desktop and /bin***

Solution:

- `ls /test | grep zip`
- `ls /test | head -n 1`
- `ls /test | tail -n 1`
- `ls /test /Desktop | sort | less`

Activity 3:

Perform the following tasks

- ***Examine the network using Ping command and view the performance statistics***

Solution

- `ping localhost`

3) Graded Lab Tasks

Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.

Task 1:

1. Use sort and unique command to sort a file and print unique values.

2. Use head and tail to print lines in a particular range in a file.

3. Use ls and find to list and print all lines matching a particular pattern in matching files.

4. Use cat, grep, tee and wc command to read the particular entry from user and store in a file and print line count.

5. Pipes the output from the cat (concatenate) process into the sort process to produce sorted output, and then pipes the sorted output into the uniq process to eliminate duplicate records.

Task 2:

1. Create a file named students.txt and insert at least 10 records.
2. Display the contents of the file using less and search for a specific keyword.
3. Use grep to:
4. Find lines containing a specific word (case insensitive)
5. Display lines that do not match a given pattern
6. Sort the file and remove duplicate records using appropriate commands.
7. Count the number of lines matching a specific pattern using pipeline commands.
8. Display the first 5 and last 5 lines of the file.
9. Use pipelines to display sorted unique records page-by-page.
10. Search all .txt files in the home directory and print lines containing a specific word.
11. Take input from the user, store it in a file, filter a keyword, and count matching lines using appropriate commands.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation

LABORATORY SKILLS ASSESSMENT (Psychomotor)
Total Marks: 100

(Max Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S< 70%	Level 3 70% ≤ S< G0%	Level 4 G0%≤ S ≤100%	Score (S)
Procedural Awareness (20)	Selects inappropriate Linux commands, shell scripting techniques, or process management methods.	Selects and applies partially appropriate Linux commands and techniques	Selects and applies considerably appropriate Linux commands and techniques.	Selects and applies completely appropriate Linux commands and techniques	
Practical Implementation (20)	Makes major critical errors in executing Linux commands, scripting, and system processes.	Makes numerous critical errors in executing commands and process management.	Makes minor non-critical errors in executing Linux commands and system operations.	Executes Linux commands and manages processes correctly with no errors.	
Process Management and Shell Scripting (20)	Program logic contains major errors with incorrect or contradictory script flow.	Program logic has some errors with occasional contradictions in process execution.	Program logic is mostly correct but may contain occasional redundancy or minor errors.	Program logic is completely correct with no contradictions or redundant processes.	
Syntax Correctness and Results (20)	Program does not follow proper syntax for Linux commands and shell scripting, leading to incorrect outputs	Program partially follows proper syntax, producing correct results for few inputs.	Program adequately follows proper syntax, producing correct results for most inputs.	Program fully follows proper syntax, producing accurate results for all inputs.	
Use of OS Tools (10)	Uses OS tools (like terminal, process manager) with limited competence.	Uses OS tools with some competence.	Uses OS tools with considerable competence.	Uses OS tools proficiently with a high degree of competence.	
Safety (10)	Requires constant reminders to follow system safety procedures (e.g., file permissions, process handling).	Requires some reminders to follow system safety procedures.	Follows system safety procedures with minimal reminders.	Routinely follows system safety procedures.	
				Marks Obtained	

LABORATORY SKILLS ASSESSMENT (Affective)
Total Marks: 40

Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S < 70%	Level 3 70% ≤ S < G0%	Level 4 G0% ≤ S ≤ 100%	Score (S)
Attitude s Engagement (5)	Shows little interest in lab activities; does not participate actively.	Participates occasionally but lacks enthusiasm and consistency.	Engages actively in most lab activities with interest.	Highly motivated, participates enthusiastically, and shows a proactive approach	
Responsibility s Punctuality (5)	Frequently misses deadlines and is often late to lab sessions.	Occasionally late or misses deadlines but tries to catch up.	Submits work on time and attends lab sessions regularly.	Always punctual, meets deadlines, and takes full responsibility for assigned tasks.	
Collaboration s Teamwork (10)	Rarely collaborates, struggles to work in a team, and does not contribute effectively.	Works with team members occasionally but struggles with communication.	Cooperates well, contributes effectively, and maintains professional interactions.	Actively engages in teamwork, supports peers, and demonstrates excellent collaboration.	
Communicatio n s Presentation Skills (10)	Struggles to explain concepts, unclear verbal/written communication.	Communicates ideas with some clarity but lacks confidence or coherence.	Presents ideas effectively with minor issues in clarity or structure.	Communicates clearly, confidently, and effectively in all aspects of lab work.	
Report Quality (10)	Report contains many errors.	Report is somewhat organized with some spelling or grammatical errors.	Report is well organized and cohesive but contains some grammatical errors.	Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished.	
				Marks Obtained	