|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Finolex Academy of Management and Technology, Ratnagiri | | | |
| **Department of Information Technology** | | | |
| **Subject:** | **Unix Lab(**SE ITL402**)** | | | |
| **Class:** | **SE IT / Semester – IV (CBCGS) / Academic year: 2017-18** | | | |
| **Name of Student:** | **Kazi Jawwad A Rahim** | | | |
| **Roll No:** | **28** | | **Date of performance (DOP) :** | **22/03/2018** |
| **Assignment/Experiment No:** | | **09** | **Date of checking (DOC) :** |  |
| **Title: To implement grep, awk, perl scripts** | | | | |
| **Marks:** | |  | **Teacher’s Signature:** |  |

**1. Aim**: To implement grep, awk, perl scripts

**2. Prerequisites**:

C Programming Language and Operating System

**3. Hardware Requirements**:

* PC with minimum 2GB RAM

**4. Software Requirements:**

* Fedora installed.

**5. Learning Objectives:**

To learn awk, grep, perl scripts.

**6.Course Objectives Applicable: LO1, LO4**

**7. Program Outcomes Applicable: PO2, PO3, PO4**

**8. Program Education Objectives Applicable: PEO2, PEO3, PEO4**

**Theory:**

**awk**

It is a scripting language which is used for manipulating the data and generating reports.

Built-in variables:

NR - It keeps the current count of the number of input records.

NF - It keeps the count of numbers of fields within the current input records.

FS - It contains the fields separator character which is used to divide the fields on input line.

**grep**

Global regular expression print. It processes the text line by line and prints line which matches the specific pattern.

**Perl**

It performs following tasks -

i) Edit file content

ii) Handle line separator.

iii) Check syntax errors.

iv) Load modules.

v) Perform wrapping.

vi) Execute perl code.

vii) Set input line separator.

viii) Split input line

Create a file employee with field

name, designation, department, salary

Insert at least 10 records.

>>>vi exp9.txt

John CEO Automobile 50000

James Worker Automobile 9800

Mark HR IT 40000

Kein Worker IT 15000

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

1] AWK

**a) Default behavior of awk.**

Description: By default the awk prints the data from specified file.

Syntax: awk '{print}' filename.extension

OUTPUT:

[students@localhost ~]$ awk '{print}' exp9.txt

John CEO Automobile 50000

James Worker Automobile 9800

Mark HR IT 40000

Kein Worker IT 15000

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

**b) Prints the lines which matches the given pattern.**

Description: It prints all the lines having matching string.

Syntax: awk '/string/ {print}' filename.extension

OUTPUT:

[students@localhost ~]$ awk '/IT/ {print}' exp9.txt

Mark HR IT 40000

Kein Worker IT 15000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

**c) Splitting a line into two fields.**

Description: It splits given line into two fields.

Syntax: awk '{print $field1,$filed3}' filename.extension

OUTPUT:

[students@localhost ~]$ awk '{print $1,$4}' exp9.txt

John 50000

James 9800

Mark 40000

Kein 15000

Marsh 5000

Flein 5000

Imran 30000

Ismat 25000

Salman 5000

Amir 5000

**d) Built-in variable in awk**

**1) Use of NR built-in variables**

Description: This command gives line no to each line.

Syntax: awk '{print NR,$0}' filename.extension

OUTPUT:

[students@localhost ~]$ awk '{print NR,$0}' exp9.txt

1 John CEO Automobile 50000

2 James Worker Automobile 9800

3 Mark HR IT 40000

4 Kein Worker IT 15000

5 Marsh Staff Automobile 5000

6 Flein Staff Automobile 5000

7 Imran Manager IT 30000

8 Ismat SubManager IT 25000

9 Salman Staff IT 5000

10 Amir Staff IT 5000

**2) Use of NF built-in variables**

Description: This command gives records of first and last field.

Syntax: awk '{print $1,$NF}' filename.extension

OUTPUT:

[students@localhost ~]$ awk '{print $1,$NF}' exp9.txt

John 50000

James 9800

Mark 40000

Kein 15000

Marsh 5000

Flein 5000

Imran 30000

Ismat 25000

Salman 5000

Amir 5000

**3) Display line number to & from.**

Description: It displays line numbers only to and from

Syntax: awk 'NR==to,NR==from {print NR,$0}' filename.extension

OUTPUT:

[students@localhost ~]$ awk 'NR==3,NR==9 {print NR,$0}' exp9.txt

3 Mark HR IT 40000

4 Kein Worker IT 15000

5 Marsh Staff Automobile 5000

6 Flein Staff Automobile 5000

7 Imran Manager IT 30000

8 Ismat SubManager IT 25000

9 Salman Staff IT 5000

**4) Count the no of line in a file.**

Description: It will result the no of lines in the file.

Syntax: awk 'END {print NR}' filename.extension

OUTPUT:

[students@localhost ~]$ awk 'END {print NR}' exp9.txt

10

**5) Printing the line more than number of characters.**

Description: It will print all the lines having more than number of characters.

Syntax: awk 'length($0)>nooflines' filename.extension

OUTPUT:

[students@localhost ~]$ awk 'length($0)>20' exp9.txt

John CEO Automobile 50000

James Worker Automobile 9800

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

**6) Find the employees having salary greater than amount**

Description: It will result the name of employees having salary greater than amount.

Syntax: awk '$4>amount' filename.extension

OUTPUT:

[students@localhost ~]$ awk '$4>5000' exp9.txt

John CEO Automobile 50000

James Worker Automobile 9800

Mark HR IT 40000

Kein Worker IT 15000

Imran Manager IT 30000

Ismat SubManager IT 25000

**7) Print the list of employees in specific department**

Description: It will result employees in specific departments

Syntax: awk '$3 ~/specific/' exp9.txt

OUTPUT:

[students@localhost ~]$ awk '$3 ~/IT/' exp9.txt

Mark HR IT 40000

Kein Worker IT 15000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

**8) awk post processing**

Description: It will add lines in the starting and ending of the file.

Syntax: awk 'BEGIN {print "The employee details are:"}

>

> {print $0}

>

> END {print "finish"}' filename.extension

OUTPUT:

[students@localhost ~]$ awk 'BEGIN {print "The employee details are:"}

>

> {print $0}

>

> END {print "finish"}' exp9.txt

The employee details are:

John CEO Automobile 50000

James Worker Automobile 9800

Mark HR IT 40000

Kein Worker IT 15000

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

finish

**2] grep**

**a) Search a string in a file.**

Description: It will result all the lines having specified string.

Syntax: grep “string” filename.extension

or grep –i “string” filename.extension

OUTPUT:

[students@localhost ~]$ grep "IT" exp9.txt

Mark HR IT 40000

Kein Worker IT 15000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

[students@localhost ~]$ grep -i "IT" exp9.txt

Mark HR IT 40000

Kein Worker IT 15000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

**b) Displaying line numbers.**

Description: It will display lines having specific string.

Syntax: grep –n “string\*” filename.extension

OUTPUT:

[students@localhost ~]$ grep -n "IT\*" exp9.txt

3:Mark HR IT 40000

4:Kein Worker IT 15000

7:Imran Manager IT 30000

8:Ismat SubManager IT 25000

9:Salman Staff IT 5000

10:Amir Staff IT 5000

**c) Display n lines before match.**

Description: It will display lines before/after/around the match.

Syntax: grep -B 2 -i "string" filename.extension

or grep -A 2 -i "string" filename.extension

or grep -C 2 -i "string" filename.extension

OUTPUT:

[students@localhost ~]$ grep -B 2 -i "IT" exp9.txt

John CEO Automobile 50000

James Worker Automobile 9800

Mark HR IT 40000

Kein Worker IT 15000

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

[students@localhost ~]$ grep -A 2 -i "IT" exp9.txt

Mark HR IT 40000

Kein Worker IT 15000

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

[students@localhost ~]$ grep -C 2 -i "IT" exp9.txt

\John CEO Automobile 50000

James Worker Automobile 9800

Mark HR IT 40000

Kein Worker IT 15000

Marsh Staff Automobile 5000

Flein Staff Automobile 5000

Imran Manager IT 30000

Ismat SubManager IT 25000

Salman Staff IT 5000

Amir Staff IT 5000

**d) Count of word.**

Description: It will result the count of specified string

Syntax: grep -c "string" filename.extension

OUTPUT:

[students@localhost ~]$ grep -c "IT" exp9.txt

6

**e) Position of match in a line.**

Description: The output is not the position instead it is the byte offset of whole file.

Syntax: grep -o -b "string" filename.extension

OUTPUT:

[students@localhost ~]$ grep -o -b "IT" exp9.txt

63:IT

84:IT

163:IT

189:IT

211:IT

230:IT

**3] Perl**

**a) Command line**

Description: It will print perl file directly via command line.

Syntax: perl -e 'print "string\n"' filename.extension

OUTPUT:

[students@localhost ~]$ perl -e 'print "IT\n"' exp9.txt

IT

**b) Printing a name**

Program:

print "Enter your name";

$name=<STDIN>;

print "Hello, ${name} .....Welcome To PERL\n";

Syntax: perl program.pl

OUTPUT:

[students@localhost ~]$ vi exp91.pl

[students@localhost ~]$ perl exp91.pl

Enter your nameJawwad

Hello, Jawwad

.....Welcome To PERL

**c) Write a perl script script program to perform all arithmetic operations.**

Program:

print "Enter two numbers\n";

$a=<STDIN>;

$b=<STDIN>;

$m=$a + $b;

$n=$a - $b;

$o=$a \* $b;

$p=$a / $b;

print "Addition=${m}\n";

print "Subtraction=${n}\n";

print "Multiplication=${o}\n";

print "Division=${p}\n";

Syntax: perl program.pl

OUTPUT:

[students@localhost ~]$ vi perl92.pl

[students@localhost ~]$ perl perl92.pl

Enter two numbers

5

4

Addition=9

Subtraction=1

Multiplication=20

Divisionion=1.25

**Learning Outcomes Achieved**

Learned awk, grep and perl scripts.

**Conclusion:**

Thus we have studied to implement awk, grep and perl scripts.

**13. Experiment/Assignment Evaluation**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **SR** | **Parameters** | **Weight** | **Excellent** | **Good** | **Average** | **Poor** | **Not as per requirement** |
| **Scale Factor ->** | 5 | 4 | 3 | 2 | 0 |
| 1 | Technical Understanding | 25 |  |  |  |  |  |
| 2 | Performance / Execution | 25 |  |  |  |  |  |
| 3 | Question Answers | 20 |  |  |  |  |  |
| 4 | Punctuality | 20 |  |  |  |  |  |
| 5 | Presentation | 10 |  |  |  |  |  |
|  | Total out of 100 -->  #(to be converted as per term-work evaluation applicable to the subject) | | **∑ (Weight \* Scale Factor)/5 = \_\_\_\_\_\_\_\_** | | | | |

**References**:

[1] Unix, concepts and applications by Sumitabha Das, McGraw-Hill

[2] Mastering Shell Scripting, Randal. K. Michael, Second Edition, Wiley Publication

**Viva Questions**

* What is awk stand for?
* What is grep stand for?
* What is perl script?