Session – 7 : Application of regex : the awk command
22-08-2019, 09:00 – 10:00 Hrs, RJN 302

Application in vi editor:

One can use the search and replace command of "sed" also in vi editor. In the command, press ":" to open the command at the bottom of the windows and enter the following string to try it out on a file like "RollList.csv".

```
:%s/^\(.*\),\(.*\)$/\2,\1@smail.iitm.ac.in/g
```

The "%" at the beginning tells that search/replace command should be applied to the whole file. If we wish to apply to first 10 lines, we would write as follows:

```
:1,10s/^\(.*\),\(.*\)$/\2,\1@smail.iitm.ac.in/g
```

The "g" at the end tells that search/replace command should be applied to the lines as many times as possible. If we wish to apply only once, we would write as follows:

```
:%s/^\(.*\),\(.*\)$/\2,\1@smail.iitm.ac.in/
```

Use of awk command

One can use awk command in two ways:

[1] The script is provided on the command line itself.
[2] The script is provided as a separate file, say, "myscript.awk"

In case [1] the command would look as follows:

```
cat RollList.csv | awk -e ‘BEGIN{print “Beginning...”; c=0;}{c+
+;}END{print c;}’
```

This command does the following. The commands within the block "BEGIN" are executed once before the lines are processed. The string "Beginning..." is printed on the screen and a variabe called "c" is initialized with a value "0".
The block that follows is executed once for each incoming line. The command given here is to increment the variable "c'.
The commands within the block END are executed once after all the lines are processed. The value of the variabel "c" is printed out.
This script counts the number of lines in the file "RollList.csv".

One can add more commands in each of these blocks as needed. The syntax is similar to C language. Refer to the documentation of the awk command for more information.

In case [2] the command would look like the following.

```
cat RollList.csv | awk -f myscript.awk
```

The option “-f” indicates that the name that follows is that of a file that contains the awk script. The contents can be the following to achieve the exact task as above example.

```
BEGIN{
      print “Beginning...”;
      c=0;
}
{
      c++;
}
END{
      print c;
}
```

An example:

The following example with documentation along with the script shows what was illustrted in the class.

```
#!/usr/bin/gawk -f
#
# This script lists student statistics for MM2090
# during July-2019 semester.
# Run this as follows:
#
# cat RollList.csv | awk -f myscript.awk
#
# This block is executed once before the lines are read and processed.
BEGIN{
      print "beginning...";
      c=0;
      mecount=0;
      mmcount=0;
      cecount=0;
      freshies=0;
      seniors=0;
      # This is a field separator to split the incoming line
      # into fields. Default separator is a blank space.
      FS=",";
}
# This block is executed once per each line read.
{
      c++;
      print $0;
      r=$1;
      n=$2;
      # This if loop checks if the roll number starts with ME
      if(r ~ /^ME/) mecount++;
      if(r ~ /^MM/) mmcount++;
      if(r ~ /^CE/) cecount++;
      # This if loop checks if the roll number has 19 in the third
      # and fourth positions
      if(r ~ /^..19/) freshies++;
      # This if loop checks if the roll number does not have 9 in
      # the fourth position
      if(r ~ /^..1[^9]/) seniors++;
      # Associative array with the roll number as index/key and
      # name of the student as value
      names[r]=n;
      # Associative array with the roll number as index/key and
      # email of the student as value
      # The function tolower() converts a string to lower case
      email[r]=tolower(r) "@smail.iitm.ac.in";
      # Associative array with the roll number as index/key and
      # a random number between 1 and 8 as value.
      # The function rand() outputs a random number between 0 and 1
      group[r] = int(1+rand()*8);
}
# This block is executed once after all the lines are read and processed.
END{
      print c;
      print "...closing";
      print("Number of mech students: ", mecount);
      print("Number of meta students: ", mmcount);
      print("Number of civil students: ", cecount);
      print("Number of freshies: ", freshies);
      print("Number of seniors: ", seniors);
      print("----------------------------------");

      # This for loop runs over all the keys of the associative
      # array called names
      for (r in names) {
            print(r, " : ", names[r], " : ", email[r], " : ", group[r]);
      }
      print("----------------------------------");
      print("Students in group 1 ");

      # This for loop runs over all the keys of the associative
      # array called email
      for (r in email) {
            # This if loop checks if the value of the associative
            # array for the key r has a value 1 meaning group 1
            if(group[r] == 1) print(r, " : ", names[r], " : ", email[r], " : ",
group[r]);
      }
}
```

Homework:

Execute this script and see the output.

Modify it to try out your understanding of the regex and code capabilities of awk.