Session – 9 : The awk command

28-08-2019, 11:00 – 12:00 Hrs, RJN 302

[1] Usage of a scriptfile for awk:

You can keep the awk script in a file and use it on command prompt as follows.

1. You can name the script file as you wish. It may be a good idea to name it with ".awk" ending, say "`myfile.awk`".
2. You can process the lines from a file like RollList.csv using the command "`cat RollList.csv | awk -f myfile.awk`"
3. First line of the script file can have "`#!/usr/bin/gawk -f`". Give executable permissions to the file using the command "`chmod u+x myfile.awk`". Now you can run the script to process the lines from a file like RollList.csv using the command "`cat RollList.csv | ./myfile.awk`"
4. The first line "`#!/usr/bin/gawk`" tells the bash shell that the program "`/usr/bin/gawk`" should be used as an interpreter for the script that follows in the file. This is a generic feature and can be used for all kinds of programs as interpreters for bash scripts. If the program name is given wrongly, bash will throw an error that the interpreter could not be found.
5. An awk script contains three sections, each enclosed using flower brackets {}. The section that starts with BEGIN gets executed before any line is read from the input. You can use this section to initialize variables. The main body is contained in the second section and has no name before the flower brackets. This gets executed once for every line from the input. That is, if the input has 1000 lines, this main body gets executed 1000 times, once per line. The section that starts with END gets executed after the input is closed. You can use this to wrap up any computation, print out things and close.
6. Warning: The script file for awk should not have anything between the three sections. The flower brackets enclosing the three sections should not contain anything between them. All comments shall be inside the sections. You can use a hash # to document your script with comments. All text after the hash till the end of the line will be ignored by the interpreter.

Contents of the script `myscript.awk` used for the classroom example are given below.

```
#!/usr/bin/gawk -f
# MM2090 : Introduction to Scientific Computing
# Dept of MME, IIT Madras, Chennai 600036 INDIA
# G. Phanikumar gphani@iitm.ac.in
#
# 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);
}
END{
# This block is executed once after all the lines are read and processed.
      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.
            if(group[r] == 1) print(r, " : ", names[r], " : ", email[r], " : ",
group[r]);
      }
}
```

Run the above script and see the output. The comments inside the three code blocks convey the purpose. Notice the similarity of the syntax for pattern matching using egrep.

Homework

[1] Create a comma separated value (`csv`) file that contains three columns of data where the second and third columns contain the square and cube of the number in the first column, respectively. You can do it manually or using a shell script. Create an awk script that reads each line and prints out the sum of all three columns as the fourth column. In the last line the average of each of the four columns shall be printed out along with a remark on how many lines have been processed.

[2] Look up the log files in `/var/log` directory of your ubuntu machine. Pick up one of those files and extract statistics from that file using awk. Every line in such a log file has a pattern that you can use to extract the fields and analyse.

[3] The command “`ls -lR`” gives a long listing of all the files in the current folder recursively. Redirect that to a file. Analyze the lines in that file to report statistics on the number of files that are within intervals such as 50 kb, 250 kb, 1 MB, 10 MB and so on. That is, report how many files have a size larger than 1 MB but less than 10MB etc.,