UNIVERSIDAD DE COSTA RICA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS DE LA
COMPUTACIÓN E INFORMÁTICA

CI-0124– Computabilidad
Prof. Juan Jose Vargas Morales

Tarea de programación 5

Elaborado por:

José Ignacio Víquez Rojas B78451

Problema # 1 (30 %)

El archivo censo2000.txt contiene datos extraídos de un censo que se llevó a cabo en Estados Unidos en 2000. Los datos mostrados corresponden a condados del estado de Washington. Descargue este archivo y examínelo para que se dé una idea de su contenido. El archivo consta de 4 columnas de datos separadas por marcas de tabulador que corresponden al nombre del condado, población, área acuática y área terrestre, ambas en millas cuadradas. La primera línea del archivo es el encabezado de las columnas.

Escriba un script AWK llamado estadisticasCondados.awk que procese el archivo y para cada condado imprima su nombre, la densidad de población por milla cuadrada terrestre, y el porcentaje de la superficie del condado que es agua. Para esto último, no olvide que el porcentaje debe calcularse sobre la totalidad de la superficie del condado (agua + tierra).

Solución:

La solución se presenta en el documento, llamado EstadisticasCondados.awk y para ejecutarlo se usa la línea:

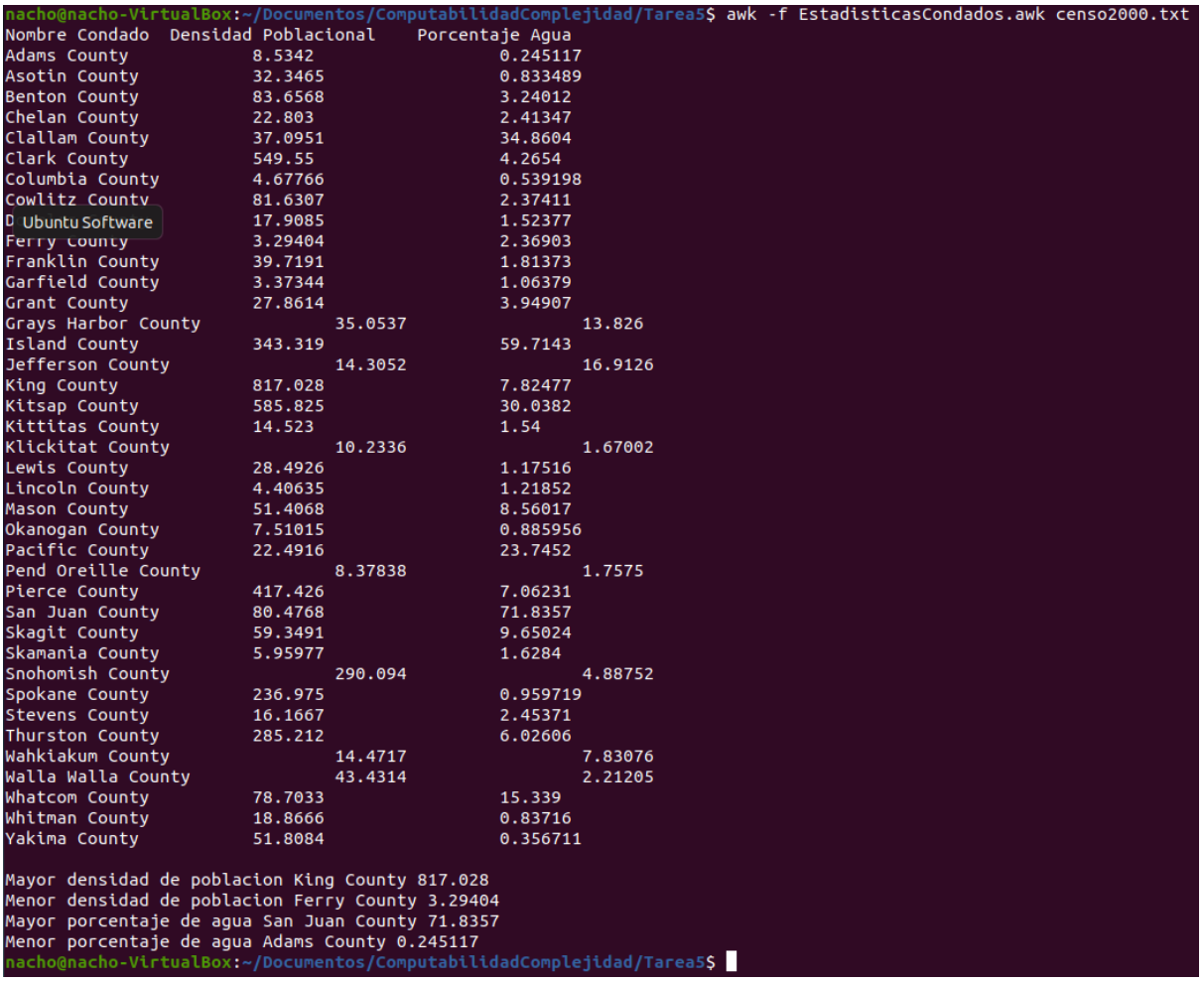
awk -f EstadisticasCondados.awk censo2000.txt

La solución primero cambia el separador de palabras por "\t" para así poder referenciar las columnas que vienen en el archivo como palabras. Después para cada línea, que representa un condado nuevo, se calculan los valores que se solicitan y se imprimen.

Para los cálculos de los mayores y menores, se crearon variables en donde se almacenan los valores de los condados que cumplen con ser los de mayores o menores valores. Cada que se lee una nueva línea se compara el valor obtenido para, por ejemplo, la densidad poblacional si este valor supera el antiguo valor de mayor densidad el valor se cambia por el recién calculado. De igual manera se realiza para la mayor densidad de agua y para los menores valores; el cambio que se realizó fue que ahora se busca por el menor valor en vez de por el mayor para realizar el intercambio.

Cuando se termina de procesar todo el archivo entonces se imprimen los valores almacenados en las variables para los mayores y menores.

Capturas del funcionamiento del script:



```
nacho@nacho-VirtualBox:~/Documentos/ComputabilidadComplejidad/Tarea5$ awk -f EstadisticasCondados.awk censo2000.txt
Nombre Condado  Densidad Poblacional    Porcentaje Agua
Adams County            8.5342                  0.245117
Asotin County           32.3465                 0.833489
Benton County           83.6568                 3.24012
Chelan County           22.803                  2.41347
Clallam County          37.0951                 34.8604
Clark County            549.55                  4.2654
Columbia County         4.67766                 0.539198
Cowlitz County          81.6307                 2.37411
D Ubuntu Software       17.9085                 1.52377
Ferry County            3.29404                 2.36903
Franklin County         39.7191                 1.81373
Garfield County         3.37344                 1.06379
Grant County            27.8614                 3.94907
Grays Harbor County             35.0537                 13.826
Island County           343.319                 59.7143
Jefferson County                14.3052                 16.9126
King County             817.028                 7.82477
Kitsap County           585.825                 30.0382
Kittitas County         14.523                  1.54
Klickitat County                10.2336                 1.67002
Lewis County            28.4926                 1.17516
Lincoln County          4.40635                 1.21852
Mason County            51.4068                 8.56017
Okanogan County         7.51015                 0.885956
Pacific County          22.4916                 23.7452
Pend Oreille County             8.37838                 1.7575
Pierce County           417.426                 7.06231
San Juan County         80.4768                 71.8357
Skagit County           59.3491                 9.65024
Skamania County         5.95977                 1.6284
Snohomish County                290.094                 4.88752
Spokane County          236.975                 0.959719
Stevens County          16.1667                 2.45371
Thurston County         285.212                 6.02606
Wahkiakum County                14.4717                 7.83076
Walla Walla County              43.4314                 2.21205
Whatcom County          78.7033                 15.339
Whitman County          18.8666                 0.83716
Yakima County           51.8084                 0.356711

Mayor densidad de poblacion King County 817.028
Menor densidad de poblacion Ferry County 3.29404
Mayor porcentaje de agua San Juan County 71.8357
Menor porcentaje de agua Adams County 0.245117
nacho@nacho-VirtualBox:~/Documentos/ComputabilidadComplejidad/Tarea5$
```

Problema # 2

Parte a) (20 %)

Escriba un script de AWK para procesar estos archivos (arenalBadLogin.txt, secure-20210418, secure-20210426, secure-20210502, secure-20210510, secure) y generar un reporte que calcule el número de veces que cada dirección IP ha intentado hacer un login, según el archivo arenalBadLogin.txt y los archivos de tipo secure.

Tome en cuenta que en los archivos "secure" se deben filtrar las líneas que contengan "Failed password" en su descripción.

La columna de "ambos" debe mostrar un * cuando la IP se encuentra en ambos tipos de archivos.

Solución:

La solución se presenta en el documento Ejericio2partea.awk, llamado y para ejecutarlo se usa la línea:

awk -f Ejericio2partea.awk badLoginArenal.txt secure secure-20210418 secure-20210426 secure-20210502 secure-20210510

Para este ejercicio se crearon 3 arreglos todos indexados por el número de la IP del ataque. Los arreglos eran :

- **arregloIPS:** Almacena las IP's de los ataques realizados.
- **arregloArenal:** Almacena la cantidad de ataques realizados que fueron detectados en el archivo de "badLoginArenal.txt"
- **arregloSecure:** Almacena la cantidad de ataques realizados que fueron detectados en los archivos de tipo *secure.*

Cuando se leía una nueva línea, para saber de cuál archivo procede la línea pasa por un "filtro" en donde según la estructura de la misma se sabe si es de los ataques del archivo "badLoginArenal.txt" o de los archivos tipo *secure.* Este filtro funciona de la siguiente manera, si la línea tiene la palabra "ssh:notty" en la segunda palabra sabemos que se trata de líneas del archivo del arenal, si no entonces es de los archivos *secure.* En este punto para las líneas del archivo del arenal podemos sumar a los ataques provenientes de está IP y guardar la suma en el campo correspondiente del arreglo **arregloArenal.**

Con las líneas de los archivos *secure* hay que aplicar otro filtro ya que solo nos interesan aquellas líneas que nos indican intentos de intrusión. Para esto si la sexta palabra dice "Failed" sabemos que se trata de un ataque. Después de esto basta con sumar un ataque más a la IP en el campo correspondiente a la IP en **arregloSecure**.

Finalmente para la fila de "ambos" basta con recorrer **arregloArenal** y **arregloSecure** y revisar si para una IP sus valores en esas entradas es mayora a o, si esto se cumple entonces ponemos un "*" de otra manera se queda en blanco.

Capturas del funcionamiento del script:

```
nacho@nacho-VirtualBox:~/Documentos/ComputabilidadComplejidad/Tarea5$ awk -f Ejericio2partea.awk badLoginAr
enal.txt secure secure-20210418 secure-20210426 secure-20210502 secure-20210510
IP              ArenalBadLogin  secure(s)       Ambos
206.130.141.138 0               3
222.212.85.136  38              38              *
205.134.173.70  1               1               *
188.166.62.86   0               4
202.188.101.106 0               22
61.132.225.82   0               23
121.5.142.223   16              9               *
178.62.102.99   12              12              *
129.226.173.71  0               5
119.45.239.10   10              10              *
45.232.73.83    1               1               *
157.122.6.2     0               8
124.239.168.74  12              12              *
106.12.219.184  1               1               *
42.200.206.52   0               3
192.187.111.130 2               1               *
167.172.145.53  0               18
42.51.9.193     0               1
223.197.186.7   0               21
123.232.42.210  0               20
49.198.210.41   40              40              *
```

Parte b) (20 %)
Mejore la calidad de la salida anterior por medio de ordenar de mayor a menor las direcciones IP con base en el número de intentos fallidos, sumando los intentos en ambos archivos. (no se trata de ordenar por dirección IP, sino por el número de intentos). En el listado debe verse cada dirección IP acompañada de la suma del número de intentos, empezando por la IP con más intentos. Para lograr esto, estudie las funciones asort y asorti de AWK. Sin embargo, como no queremos contar un mismo intento por duplicado, la suma debe calcularse como dos intentos, sólo si la fecha y hora difieren en más de 1 minuto. Para ello hay que restar las fechas y horas, tomando su valor absoluto y compararlo contra un número dado de minutos, digamos 1. Si difieren en más de 1 minuto sumamos 2, si no, sumamos 1. Para hacer este cálculo, es mejor usar una función definida por el usuario que reciba las dos fechas y horas y devuelva la diferencia.

Solución:
La solución se presenta en el documento Ejericio2parteb.awk, llamado y para ejecutarlo se usa la línea:

gawk -f Ejericio2parteb.awk badLoginArenal.txt secure secure-20210418 secure-20210426 secure-20210502 secure-20210510

Para está parte del ejercicio se usó el mismo “filtro” que se usó en la parte a) basado en buscar la palabra “ssh:notty” para diferenciar si las líneas son del archivo “badLoginArenal” o de los archivos de *secure.*
Se crearon los siguientes arreglos:

- **arregloIPS:** Almacena las IP’s de los ataques realizados.
- **arregloTiempoArenal:** Almacena la cantidad de ataques realizados para una IP pero no los almacena con una cuenta, los almacena con un string. Este arreglo está indexado con la IP de donde provienen los ataques, y los valores de cada entrada es un string que tiene la siguiente manera :

  HoraAtaque1---HoraAtaque2---HoraAtaque3

  Como podemos ver el string muestra cada hora de realización de un ataque separado por tres guiones.

La explicación se va a realizar desde dos perspectivas, la de los ataques del archivo del arenal y la de los ataques de los archivos *secure.*

Ataques encontrados en el archivo " badLoginArenal.txt":
Para contar estos ataques bastaba con recorrer el archivo y cada que se encuentra un nuevo ataque para una IP concatenar la hora de realización del ataque a la entrada de está IP en **arregloTiempoArenal**.

Ataques encontrados en los archivos *secure:*
Cuando se detecta un ataque proveniente de una IP que no tiene ataques en su entrada de **arregloTiempoArenal**, solo agregamos el ataque al string de la entrada. Por otro lado si la IP si tiene ataques en su entrada del arreglo hemos de revisar si el nuevo ataque ya está en el arreglo mediante la comparación de la hora de realización del ataque. Para esto se realiza un llamado a una función llamada "compararFechas" en donde se comparan todas las fechas de la entrada del **arregloTiempoArenal** para la IP correspondiente con la fecha del nuevo ataque.
La comparación se basa en lo siguiente, existe una bandera que puede tomar valores de 0,1. Si la fecha nueva es igual a alguna de las fechas del string entonces la bandera se cambia a 1, indicando que el nuevo ataque es repetido. Si al comparar con todas las fechas de ataques de la entrada del arreglo la bandera queda en 0 quiere decir que el ataque es nuevo y en ese caso lo agregamos al string de está IP.
El método regresa el string modificado o no para la entrada de la IP, esto se asigna al arreglo y se continúa con la siguiente línea.
Cuando ya se terminó de leer todos los archivos se realiza un conteo de los ataques para cada IP, y está conteo se almacena en un arreglo llamado **arregloCantidadAtaquesFinal**. Por último se usó la función asorti para ordenar un arreglo que se usa después para indexar las salidas en orden.

Capturas del funcionamiento del script:

```
nacho@nacho-VirtualBox:~/Documentos/ComputabilidadComplejidad/Tarea5$ gawk -f Ejericio2parteb.awk badLoginA
renal.txt secure secure-20210418 secure-20210426 secure-20210502 secure-20210510
IP              CantidadAtaques
222.252.30.29   398
129.226.113.50  284
176.122.158.234 233
139.59.18.197   230
181.126.83.37   228
140.238.206.139 222
211.173.58.253  221
202.115.29.234  212
111.229.156.53  203
95.217.187.253  200
88.255.102.45   200
85.244.244.81   200
82.196.9.161    200
5.189.221.17    200
46.101.249.232  200
46.101.189.37   200
41.191.116.18   200
35.245.33.180   200
201.48.36.33    200
200.123.130.39  200
192.241.173.142 200
139.155.127.59  200
13.65.16.18     200
122.165.149.75  200
```

Parte c) (30 %)

Ahora se debe agregar información del sitio IP tal como país, ciudad, nombre de la red. Para ello, experimente por separado con el comando whois de Linux. Lea el manual en línea o una guía de su uso. Considere la siguiente ayuda.

En AWK se pueden ejecutar comandos del sistema operativo de dos maneras: mediante la llamada system(), o mediante la construcción de una hilera que contiene el comando a ejecutar. La salida del comando en la hilera se puede pasar por un pipe a la función getline que la captura y la guarda en una variable. Posteriormente, esa variable se puede utilizar dentro de AWK para cualquier propósito, como imprimirla junto con otros datos relacionados. Investigue todo esto.

Solución:

Para está parte se usó como código base el generado en la parte b), y al final se le agregó la funcionalidad del comando “whois”. Para usar esto se recorre el **arregloIPS** y para cada una de las entradas del arreglo se llama el comando “whois” para obtener información al respecto. Como este comando retorna mucha información adicional a la que ocupamos entonces se usaron expresiones regulares para localizar las líneas de salida que se ocupan imprimir. Estas expresiones regulares son :

- **[Cc]ountry:** Para identificar la información del país de procedencia del ataque.
- **[Cc]ity:** Para identificar la información de la ciudad de donde procede el ataque.

- **[Nn]et[Nn]ame:** Identifica la información referente al nombre de la red de donde procede el ataque.
- **[Aa]ddress:** Identifica la localización de donde procede el ataque. Se puso está opción ya que en muchas ocasiones el parámetro de “city” no se presenta en la llamada de “whois” y con la indicación del “address” se puede obtener está información.

Capturas del funcionamiento del script: