Supuesto Práctico 1

Alejandro Del Hierro Diez

7 de marzo de 2019

UNIVERSIDAD DE VALLADOLID
Gramáticas y Lenguajes Formales
Curso 2018-2019

Índice

1. Introducción

En el siguiente informe se explicará todo lo relativo a los programas y ficheros necesarios para la elaboración de los programas exigidos en este primer supuesto práctico y los listados finales creados a partir de estos programas.

2. Secuencia de ejecución

En el directorio se encuentran los programas elaborados, los cuales son: `guion, ecsv2csv, campos, campo2num, opc1, listado y convfecha`. A continuación se resume el funcionamiento de dichos programas, ordenados segun la secuencia de ejecución:

1. `guion`: Script ejecutable bash cuya ejecución realizará todo lo requerido en la practica. Desde dentro de este se hace la llamada a los siguientes.
2. `ecsv2csv`: Script ejecutable bash que contiene dos instrucciones `sed` para la normalización del archivo inicial de datos, ya que este tiene saltos de linea dentro de campos, lo cual altera la correcta lectura del fichero `.csv`.
3. `campos`: Script ejecutable awk que obtiene una lista de campos y el número correspondinte de un fichero `.csv` pasado como argumento.
4. `campo2num`: Script ejecutable bash que recibe una expresión regular y un fichero opcional y desde dentro llama a l programa `opc1`
5. `opc1`: Script ejecutable awk que a partir de una expresion regular dada encuentra el numero de campo del que se trata o con el que hay una coincidencia.
6. `listado`: Script ejecutable awk que recibe una expresión regular y un campo y a partir de esto genera un listado con las especificaciones del enunciado de la práctica.
7. `convfecha`: Script ejecutable bash que realiza la conversión de las fechas de un fichero dado mediante una instrucción `sed`.

A contiuación se explican con detalle cada uno de estos programas y las expresiones regulares implicadas.

3. Detalle de los programas

3.1. ecsv2csv

Este programa contiene dos instrucciones sed:

```
sed -i '1d' CataDatosJCyL.csv
sed -i ':x /;/ {N;s/[^;]\n//g; bx}' CataDatosJCyL.csv
```

La primera instrucción elimina la primera linea del fichero `.csv`. La segunda linea tiene un funcionamiento especial:

1. `:x` Crea una etiqueta.
2. `/;/` Expresión regular que buscará un punto y coma en los registros. Cuando lo encuentre, la añadirá esa línea al búffer de patrones.
3. `N` Añade la siguiente linea de la coincidente al búffer.
4. s/`[\^;]\n`//g Susituye el patrón por la cadena vacía. El patrón a encontrar se indica con la expresión regular `[\^;]\n`, la cual significa: "saltos de linea que no vayan precedidos de un punto y coma". Así, borrará todos los saltos de línea que no sean los de final de registro.
5. `bx` Salta a la etiqueta para empezar de nuevo, hasta acabar el fichero.

Al final de este script, se copiará el resultadp a CataDatosCSV.csv.

3.2. campos

Contiene el siguiente código awk:

```
#!/usr/bin/awk -f
BEGIN {FS=DEL;maxnf=0}
NR==1{
        for (nf=1;nf<=NF;nf++) {
              cname[nf] = $nf ;
        }
        if(NF>maxnf) maxnf = NF;
}
END{
        for (i=1;i<maxnf ;i++) {
                print i ":" cname[i];
        }
}
```

A grandes rasgos, este programa recorrerá el primer registro (NR==1) recogiendo el valor de cada uno de los campos. Así guardará el nombre en el vector `cname` -en cada posición guarda el valor del campo ($nf)-.
Finalmente impimirá un número y el valor del vector en el que se recogieron los nombres de los campos.
Como ejemplo de llamada a este programa:

```
./campos -v DEL=";" data/CataDatosCSV.csv > data/CataDatosCSV.campos
```

Con -v se le pasa el valor a la variable `DEL` (delimitador del fichero). El siguiente arugumento es el fichero con el que trabajará el programa `campos`. Finalmente se vuelca el resultado en el fichero CataDatosCSV.campos.

3.3. campo2num

Este script sólo contiene la siguiente llamada al programa awk `opc1`.

```
./opc1 -v RE="$1" ${2:-./data/CataDatosCSV.campos}
```

`opc1` recibirá la variable RE, que es una expresión regular, a la cual se le pasa el valor "$1", lo cual significa el primer argumento que se le pasa a `campo2num` cuando se le llama. El segundo argumento indica que por defecto tomará ese archivo de datos, a menos que se pase alguno en la llamada a campo `campo2num`. Un ejemplo de llamada:

```
./campo2num 'Enlace al.*'
```

Esta expreión regular buscará el patron `Enlace al` seguido de cualquier cosa (.) cero o más veces (*).

3.4. opc1

El código de este programa es el siguiente:

```
#!/usr/bin/awk -f
BEGIN {FS=":";none=0}
$2~RE{
        c1=$1;
        c2=$2;
}
""~RE{
        none=1;
}
END{
        if(none==1){
                print -1;
        }else if (c2==""){
                print 0;
        }else
                print c1;
}
```

El funcionamiento de este programa es el siguiente:

1. `FS y none`: Indica que el separador del fichero son ":", y none es una variable para posterior uso.

2. `$2-RE`: Si el campo dos tiene coincidencias con la expresión regular que se le pase, guardará en c1 el valor del campo 1 (numero de campo) y en c2 el de c2 (nombre del campo).

3. `-RE`: Si la expresión regular esta vacía, es decir, no es ninguna, pone none a 1.

4. `END`: Si none==1, imprime -1; si c2==, es decir, no ha encontrado ninguna coincidencia, imprime 0, y si no, imprime c1.

3.5. listado

Como ejemplo de lamada al programa:

```
nsec=$(./campo2num 'Sector')
./listado -v cn=${nsec} ereg=.*[Ss]alud.* data/CataDatosCSV.csv
```

`nsec` es una variable del sistema a la que se le da el valor del campo al que corresponda la expresión regular 'Sector'. A `listado` se le pasará el valor de esa variable, y una expresión regular, la cual es `.*[Ss]alud.*`. Esta expresión regular encontrará patrones que coincidan con: cualquier cosa (.) cero o más veces (*), el símbolo 'S' o el 's' una vez seguido de 'alud', y todo seguido de cualquier cosa cero o mas veces (.*). Esto es, cualquier cosa que contenga la palabra salud, ya empiece por mayúscula o por minúscula. El código de este programa es:

```
BEGIN {FS=";"}
NR==1{next;}
$cn~ereg{
        gsub("\"","");
        print $1 ": (" $14 ")";
        print "  " $cn "\n";
}
END{}
```

Para el primer registro no hace nada, así omitimos imprimir la cabecera. Si el valor del campo que se le pasa en cn ($cn) coincide (en cualquiera de los registros) con la expresión regular que se le pasa, sustituye las comillas por ninún simbolo con la orden `gsub`; imprime el primer campo (nombre del conjunto de datos) y el campo 14 (fecha de última actualización). Finalmente imprime el campo que se le pasa en cn en la linea de debajo.

3.6. convfecha

Este prgrama se usa como filtro para los listados creados con el programa anterior mediante una tubería. Únicamente ejecuta una instrucción `sed` como la siguiente:

```
sed -E 's,\(([0-9]{4})([0-9]{2})([0-9]{2})\),[\3-\2-\1],g;
s/-01-/-ENE-/g; ...'
```

Su funcionmiento es:

1. -E indica que trate expresiones regulares extendidas.

2. `s,\(([0-9]{4})([0-9]{2})([0-9]{2})\),[\3-\2-\1],g` Sustituye las partes del patron que están fuera de los grupos. Los grupos son los contenidos entre paréntesis, es decir, hay tres grupos: `([0-9]{4}),([0-9]{2})` y `([0-9]{2})`. Este fragmento de código, buscará un patron de 3 grupos, de cualquier número del 0 al 9, de 4, 2 y 2 cifras seguidos. Fuera de los 3 grupos se ven dos paréntesis de la forma `\(...\)`. Estos paréntesis escapados con `\`, son los dos paréntesis carácter que envuelven a la fecha en el fichero. De esta manera, encontrará patrones del tipo (20140611), y dividirá los dígitos en 3 grupos.
   La siguiente orden, `[\3-\2-\1]`, colocará el grupo 3 primero seguido de un guión, despues el 2 seguido de guión, y finalmente el 1, todo ello entre los corchetes que ahí se aprecian. La g indica que se haga en todas las ocurrencias de la línea. Los dos paréntesis no incluidos dentro de los grupos de la primera parte serán eliminados. De esta forma se consigue una fecha con formato [11-06-2014].

3. Lo que sigue a la primera orden de sustitución, es otra orden de sustitución que busca el patron -01- y lo sustituye por -ENE-. Esta orden hará lo mismo para todos los meses, para conseguir un formato [11-JUN-2014].

3.7. guion

Este script sólo incluye llamadas a los programas. Incluye las siguientes órdenes:

```
#!/bin/sh
#Descarga del primer fichero
curl https://datosabiertos.jcyl.es/web/jcyl/risp/es/ciencia-
tecnologia/general/1284166186527.csv > ./data/CataDatosRaw.csv

#Transformación del fichero a formato UTF-8 y envío a CataDatosJCyL.csv
iconv -f ISO-8859-15 -t UTF-8 ./data/CataDatosRaw.csv >
    ./data/CataDatosJCyL.csv

#Normalización del archivo
./ecsv2csv

#Creacion del fichero con los campos y el número que tiene cada uno
./campos -v DEL=";" data/CataDatosCSV.csv > data/CataDatosCSV.campos

#Creacion de listados finales y filtrado de la fecha mediante tubería
nurl=$(./campo2num 'Enlace al.*')
./listado -v cn=${nurl} ereg=.* data/CataDatosCSV.csv | ./convfecha >
data/CataListURLs.list

#Busca el número del campo Sector
nsec=$(./campo2num 'Sector')
```

```
./listado -v cn=${nsec} ereg=.* data/CataDatosCSV.csv | ./convfecha >
data/CataListSectores.list

./listado -v cn=${nsec} ereg=.*[Ss]alud.* data/CataDatosCSV.csv | ./convfecha >
data/CataListSectorSalud.list
```

Referencias

[1] Descripción de la orden sed (ComputerHope), *https://www.computerhope.com/unix/used.htm*

[2] Tutorial de sed (Bruce Barnett),*http://www.grymoire.com/Unix/Sed.html*

[3] Tutorial de awk, *http://www.grymoire.com/Unix/Awk.html*

[4] Descripción de la orden awk (ComputerHope) , *https://www.computerhope.com/unix/uawk.htm*