Práctica P1

Procesamiento de ficheros CSV

28 de febrero de 2019

Resumen

Este supuesto práctico representa la primera entrega práctica evaluable de la asignatura y plantea la creación de una colección de herramientas, basadas en `grep`, `sed`, `awk` y `bash` scripts, orientadas al procesamiento de ficheros de texto usando como base expresiones regulares. Los ficheros de entrada serán ficheros de texto en formato CSV que contienen datos abiertos y a partir de los cuales hay que obtener versión normalizada, extraer parte de la información y elaborar informes.

1. Enunciado

En el directorio de trabajo que elija, comience por crear un directorio con su NIF (letra final en mayúscula) (e.g. 12345678X) en el que deberán contenerse todos los ficheros asociados a este supuesto y al que, en lo que sigue, nos referiremos como directorio `<base>`.

Los diferentes pasos que se describen a continuación deberán estar contenidos en un script en `bash` que deberá llamarse `guion` y deberá ser ejecutable para el usuario propietario. La ejecución de este script permitirá ejecutar todos y cada uno de los pasos que se describen a continuación, generando todos y cada uno de los ficheros intermedios que se mencionan o sólo aquéllos que no hayan sido generados hasta ese punto (se valorará esta variante de solución).

Dentro del directorio `<base>` deberá crearse (si no existe) un directorio `data` en el que se irán descargando o generando los diferentes ficheros que se crean y manejan en este supuesto.

1.1. Descarga de catálogo

En la web de datos abiertos de la Junta de Castilla y León (JCyL en lo sucesivo) se pueden encontrar numerosos conjuntos de datos de libre acceso ordenados por áreas temáticas. La relación de todos ellos está contenida en un *Catálogo de Ficheros de Datos Abiertos* que está incluído en el propio catálogo y que se puede acceder directamente para descarga en la dirección (https://datosabiertos.jcyl.es/web/jcyl/risp/es/ciencia-tecnologia/general/1284166186527.csv).

Consulte la orden `curl` para incluir en `guion` la descarga del catálogo de datos abiertos de la JCyL desde la URL anterior al fichero de salida `CataDatosRaw.csv`, que deberá localizarse en el subdirectorio `data` como se ha indicado al principio. El fichero que descargue es un fichero CSV[1] cuyos campos están delimitados por el símbolo " (*double quotes*). Verá que hay algunos campos que se extienden a lo largo de varias líneas.

[1] Comma Separated Values

1.2. Cambio del juego de caracteres a UTF

El fichero `CataDatosRaw.csv` está codificado en el juego de caracteres `ISO-8859-1` (puede usar la orden `file` para comprobarlo). Dado que el juego de caracteres que seguramente tendrá declarado como valor por defecto será `UTF-8`, se precisa convertir el fichero anterior en otro, que deberá llamar `CataDatosJCyL.csv` (siempre en el directorio `data`). Para hacerlo, consulte la orden `iconv`.

1.3. Normalización del fichero CSV

Inspeccione el fichero `CataDatosJCyL.csv` y verá que contiene una primera línea en la que se indica la fecha de última actualización y que debemos ignorar a efectos de interpretar la estructura de lo que sigue.

Además, la segunda línea contiene una relación de nombres de campos separada por `;` que describe los contenidos de cada uno de los registros que aparecen el fichero.

Finalmente, comprobará que, como se ha indicado más arriba, hay algunos campos que contienen texto (`HTML`) que se extiende a varias líneas.

Por todo ello, en este paso se pide que **escriba un programa en `sed` para normalizar este fichero** de forma que sea más sencillo su procesamiento posterior. El programa `sed` deberá llamarse `ecsv2csv` y realizará las acciones necesarias sobre el fichero que se indique como argumento a la hora de ejecutarlo para asegurar:

- La eliminación de la primera línea de la entrada.
- La eliminación de todos los saltos de línea que no supongan cambio de registro. En otras palabras, se trata de normalizar cada registro a una secuencia de campos separados por (como ya lo están de hecho) y que, todos y cada uno, seguirán delimitados por comillas dobles.

Dentro del programa `guion` se incluirá la ejecución del programa `ecsv2csv` aplicado al fichero de entrada `CataDatosJCyL.csv` y volcando la salida en `CataDatosCSV.csv`

1.4. Obtención de la lista de campos

El fichero ya normalizado `CataDatosCSV.csv` contendrá una lista de campos (primer registro) que contiene 16 elementos, aunque el último está vacío. Deberá escribir un programa en `awk`, denominado `campos` que, dado un fichero de entrada CSV separado por proporcionado como argumento, muestre la relación de campos que se contiene en el primer registro. Cuando ejecute este programa `awk` sobre el fichero de entrada `CataDatosCSV.csv`, deberá generarse una salida como la que puede verse en la Figura 1.

El programa `campos` se ejecutará desde el script `guion` tomando como entrada el fichero `CataDatosCSV.csv`, filtrando la salida con una orden `sed` que **elimine la última línea de la entrada** y redirigiendo la salida al fichero `CataDatosCSV.campos`.

1.5. Obtención del número de campo a partir del nombre

En este siguiente paso, deberá escribir un programa `bash` denominado `campo2num` que, partiendo de una expresión regular (primer argumento de la línea de órdenes) y, opcionalmente,

```
01 : Título
02 : Descripción
03 : Tag
04 : FechaCreacion
05 : FechaUltimaActualizacion
06 : Frecuencia de actualización
07 : Sector
08 : Organismo responsable
09 : AmbitoGeografico
10 : Licencia
11 : Formatos disponibles para descarga
12 : EnlaceMasInformacion
13 : Identificador
14 : ultimaActualizacion
15 : Enlace al contenido
16 :
```

Figura 1: Relación de campos que deberá mostrar el programa `campos`, escrito en `awk`.

del nombre de un fichero (si no se especifica, deberá usarse `CataDatosCSV.campos`), muestre en la salida estándar el número de campo a que corresponde aquel que obedece al patrón representado en la expresión regular (o `0` si no hubiese ninguno y `-1` si no se especifica ninguna expresión regular como primer argumento). En la Figura 2 aparece un ejemplo de comprobación de la ejecución de este programa.

```
$ tfn=$(../campo2num 'Títu.*')
$ ffn=$(../campo2num '.*Crea.*')
$ ufn=$(../campo2num 'ultima.*')
$ lfn=$(../campo2num '.*Enlace .*')
$ echo "Titulo:␣$tfn,␣Creado:␣$ffn,␣UltMod:␣$ufn,␣Link:␣$lfn"
Titulo: 01, Creado: 04, UltMod: 14, Link: 15
```

Figura 2: Ejemplo de ejecución de `campo2num` en `bash`.

1.6. Generación de algunos listados

En este siguiente paso, deberá escribir un programa `awk` denominado `listado` que, partiendo de la asignación en línea de órdenes de un número de campo y una expresión regular opcional a sendas variables internas `awk`, genere un listado del fichero de entrada que se proporciona como argumento en la línea de órdenes. El listado deberá contener los siguientes elementos, extraídos de los campos del fichero de entrada, siempre eliminando previamente los delimitadores comilla doble:

- Nombre del conjunto de datos, seguido de ':'.
- Fecha de última modificación del conjunto de datos, de acuerdo con el valor del antepenúltimo campo de cada registro[2], encerrada entre paréntesis.
- En línea aparte con 2 blancos de indentado, el valor del campo cuyo número se corresponde con el valor numérico aportado como valor de la varible `awk` en `listado`, que habrá sido obtenido a partir del nombre correspondiente usando `campo2num`.

[2] Como podrá comprobar, hay registros que contienen, por error, más campos de los declarados en el primer registro.

La salida de este programa se filtrará a través de un programa `sed` (que se llamara `convfecha`) que convierta cada fecha de última modificación que aparece en el listado generado por `listado` al formato estándar siguiente: `DD-MMM-YYYY`, cambiando los paréntesis por corchetes (e.g. `(20140226)` se convertirá en `[26-FEB-2014 ]`). La figura 3 muestra un ejemplo de uso del programa para el fichero de entrada preparado en la sección 1.3.

```
$ ‘pwd‘
data
$ nURL=$(../campo2num 'Enlace al.*')
$ ../listado -v cn=${nURL} ereg='.*datosabiertos.*' CataDatosCSV.csv | ../convfecha
Calificaciones de Zonas de Agua de Baño: []21-DIC-2018]:
  https://datosabiertos.jcyl.es/web/jcyl/set/es/salud/calificaciones-aguas-{...}
Actividad Analítica de los Laboratorios de Control Oficial: [21-DIC-2018]:
  https://datosabiertos.jcyl.es/web/jcyl/set/es/salud/laboratorios-control-{...}
Actividad Inspectora en los Mataderos:
    [21-DIC-2018]:https://www.overleaf.com/project/5c76bfecae11c26931b12496
  https://datosabiertos.jcyl.es/web/jcyl/set/es/salud/actividad-inspectora-{...}
  {...}
$
```

Figura 3: Ejemplo de ejecución de `campo2num` en `bash`.

Usando este programa, genere los siguientes listados de conjuntos de datos (los nombres que aparecen entre corchetes son los que deberán tener los ficheros a que se volcará cada uno de ellos):

- [**CataListURLs.list**]: Listado que muestre la URL de descarga de cada conjunto.
- [**CataListSectores.list**]: Listado que muestre el 'Sector' de cada conjunto.
- [**CataListSectorSalud.list**]: Listado que muestre los conjuntos del sector salud.

2. Documentación de la práctica

Además de la documentación de código por medio de comentarios suficientes y precisos, se deberá elaborar un documento (`readme.pdf`) en el que se describa cada uno de los programas realizados en este supuesto. Se deberá describir especialmente cada expresión regular que se emplee. Se podrá usar cualquier editor de textos que se desee y se incluirán los ficheros fuente elaborados, además del `PDF` final, en el directorio `doc` dentro del `<base>`.

3. Normas de entrega

Se generará un fichero `ZIP` que contenga el directorio `<base>` y todos sus contenidos. La figura 4 ilustra la estructura directorios que debería obtenerse antes de generar el fichero comprimido. El fichero final tendrá como nombre el NIF del estudiante (e.g. `12345678X.zip`) y se subirá como entrega de la actividad P1 en el campus virtual, en los plazos indicados en dicha actividad.

```
12345678X
├── campo2num
├── campos
├── convfecha
├── ecsv2csv
├── guion
├── listado
├── ...(ficheros de apoyo opc)
├── data
│   ├── CataDatosRaw.csv
│   ├── CataDatosJCyL.csv
│   ├── CataDatosCSV.csv
│   ├── CataDatosCSV.campos
│   ├── CataListURLs.list
│   ├── CataListSectores.list
│   ├── CataListSectorSalud.list
│   └── ...
└── doc
    ├── ... (fuentes del documento)
    └── readme.pdf
```

Figura 4: Contenidos de la carpeta a entregar.

4. Evaluación

Este supuesto representa un 10 % de la calificación final de la asignatura (1 punto sobre 10). El 60 % de la calificación de esta entrega se asociará a la elaboración de la solución y el 40 % restante a la documentación de la misma. El profesor podrá requerir a cualquier estudiante aclaraciones sobre la entrega realizada antes de emitir la calificación, si lo estima conveniente. Cualquier evidencia de plagio supondrá la obtención de una calificación de '0' puntos en esta entrega.