**Cátedra:** Osvaldo Clua  
**Fecha:** Primera Oportunidad - Segundo Cuatrimestre 2007  
**Día:** 06/11/2007

**Enunciado**

Top of Form

Bottom of Form

**Expresiones Regulares**

Se pide realizar un script que reciba por parámetro un número de patente de un rodado y muestre por salida estándar la marca junto con el nombre y número/s de teléfono de su dueño. ni contamos con un archivo llamado RODADOS.dat del que solo sabemos lo siguiente:

* Sus campos estan separados por el carácter ':'.
* El segundo campo contiene el número de patente.
* El quinto campo contiene el CUIT del único dueño.
* El séptimo campo contiene la marca del rodado

También contamos con un archivo llamado TELEFONOS.dat del que solo sabemos lo siguiente:

* Sus campos están separados por el caracter ';'
* El tercer campo contiene el CUIT.
* El quinto campo contiene nombre.
* El sexto campo contiene el número de teléfono

**Importante:** Solo se pueden utilizar los comandos grep, sed, wc y echo

Top of Form

Bottom of Form

**PERL**

Desarrollar un comando PERL denominado parcial.pl que permita procesar los archivos con novedades de compras y grabar el monto acumulado en un archivo de salida.

*Se pide:*

1. Procesar todos los archivos de novedades de compras que encontramos en el directorio corriente y que cumplan con la máscara indicada en el **primer parámetro** de la invocación.
2. Encontrar el monto de la compra según sea la posición indicada en el **segundo parámetro** de la invocación.
3. Acumular el monto por artículo empleando una estructura hash y usando al “artículo” como clave y el monto como valor. El campo artículo es el campo inmediato siguiente al monto.
4. Grabar la salida o mostrar el resultado por pantalla según lo indicado en el **tercer parámetro** de la invocación.

*Input:*

* Novedades: archivos con información de compras, formato : desconocido, separado de campos: coma.
* Maestro de Artículos: archivo MaeArt, formato : artículo:descripción.

*Validaciones a efectuar:*

1. Validar que se invoque al programa con al menos 2 parámetros, si hay error mostrar por pantalla un mensaje descriptivo y terminar el programa.
2. No efectuar ningún tipo de validación especial de la máscara *(Parámetro 1: Obligatorio: nombre del archivo de input o máscara)* , emplearla tal cual se la recibe.
3. Lo que se debe validar es que se obtenda al menos un archivo de novedades. Si ningún archivo cumple la condición, mostrar por pantalla el mensaje “sin novedades” y terminar el programa.
4. Validar que la ubicación *(Parámetro 2: Obligatorio : Ubicación del campo monto)* sea un número >= 1 y < a la cantidad total de campos del resgistro de entrada. Esta validación efectuarla solo para el primer registro de un archivo. Si la ubicación queda fuera del rango, entonces mostrar un mensaje descriptivo y pasar al siguiente archivo.

*Salida.*

* Si el parámetro 3 fue informado *(Parámetro 3: opcional : Nombre del archivo de salida)* , grabar en ese archivo Artículo, monto acumulador (el separador de campos debe ser ','. Si el archivo no existe, crearlo. Si ya existe, agregar los nuevos registros.
* Si el parámetro 3 no fue informado, mostrar por pantalla: artículo:descripción - monto acumulado. artículo:descripción se obtiene del maestro de artículos MaeArt.

Top of Form

Bottom of Form

**BASH**

Se tiene un directorio donde diariamente se reciben archivos con las ventas de varias sucursales, los archivos son tel tipo vtasuc.txt, donde suc es el número de sucursal, (ej vta524.txt) el formato de los mismos es:

**Fecha Operación (aaaammdd); Cod Producto ; Cod Cliente ; Cantidad ; Importe**

Ej:

20060201;K13;457887;105;100;50

Se cuenta además con un archivo de sucursales (**/etc/SUCURSALES.txt**):

Cod Sucursal; Descripción Sucursar; Habilitada (S/N).

Se pide realizar un script que procese la información de las sucursales y de acuerdo a la opción ingresada muestre el resultado por salida standar.

**-C <código de cliente>** Mostrar monto total de las ventas de ese cliente (monto = importe x cantidad).

Validar para cada archivo que la sucursal exista en el archivo de sucursales y que este habilitada, si no es así rechazar el archivo(moverlo a un directorio de rechazados) e indicarlo por salida estándar. Validar también que la opción ingresada sea -C y que el codigo de cliente ingresado exista en el archivo de clientes **/etc/clientes.txt**. En caso de que los parámetros sean erróneos mostrar con ejemplo la correcta invocación del script.

Top of Form

Bottom of Form

**Resolución**

**EXPRESIONES REGULARES**

#!/bin/bash

###################################################################################

#HIPOTESIS Y PRECONDICIONES: #

#Un solo dueño puede tener varios teléfonos #

#Los números de teléfono están listados en lineas separadas de CUIT coincidente #

###################################################################################

# $1 = num de patente del rodado

# resultado: MARCA, NOMBRE, TEL/S

RODADO=`grep ^[^:]\*:$1: RODADOS.dat`

CUIT=`echo "$RODADO"|sed s/'^[^:]\*:[^:]\*:[^:]\*:[^:]\*:'//|sed s/':.\*$'//`

MARCA=`echo $RODADO|sed s/"^\([^:]\*:\)\{6\}"//|sed s/":.\*"//`

LINEASTELEFONOS=`grep "^\([^;]\*;\)\{2\}$CUIT" TELEFONOS.dat`

NOMBRE=`echo $LINEASTELEFONOS|sed s/'^[^;]\*;[^;]\*;[^;]\*;[^;]\*;'//|sed s/';.\*'//`

TELEFONOS=`grep "^\([^;]\*;\)\{2\}$CUIT" TELEFONOS.dat|sed s/"^\([^;]\*;\)\{5\}"//|sed s/";.\*$"//`

# \E[34m = texto azul

# \E[30m = texto negro

# Uso -e para que me tome las secuencias de escape

echo -e "\E[34mMarca: \E[30m$MARCA"

echo -e "\E[34mNombre: \E[30m$NOMBRE"

echo -e "\E[34mTelefonos: \E[30m\n$TELEFONOS"

**BASH**

#!/bin/bash

###################################################################################################

# FI.UBA.AR 75.08

# Parcial 20071106\_1 - Ejercicio BASH

# Resolucion

# Autor: Maximiliano Milicich (mmilicich)

#

# HIPOTESIS Y PRECONDICIONES:

# - Las cantidades y precios (importes) son ENTEROS (no manejamos aritmetica decimal)

# - Los archivos invalidos, por ahora no los movemos a ningun lado: solo emitimos el warning

# - El script se debe ejecutar "parado" en la ubicacion del mismo

# (working directory == ubicacion del script). Pues sino, no se encuentra el directorio "data"

###################################################################################################

# Variables y Constantes:

ARCH\_SUC="./SUCURSALES.txt"

ARCH\_CLI="./clientes.txt"

DATADIR="./data"

# validamos los argumentos recibidos:

if [ $# -ne 2 ] ; then

echo "$0 Error: Parametros incorrectos (Modo de uso $0 -C <codigo de cliente>)"

exit 1

fi

if [ $1 != '-C' ] ; then

echo "$0 Error: Parametros incorrectos (Modo de uso $0 -C <codigo de cliente>)"

exit 1

fi

# Codigo de Cliente a procesar:

COD\_CLI="$2"

# Validamos codigo de cliente:

if [ -z $(cat "$ARCH\_CLI" | grep "$COD\_CLI") ] ; then

echo "$0 Error: Codigo de Cliente $COD\_CLI invalido"

exit

fi

# Procesamos los archivos de vtas de sucursales:

MONTO\_TOTAL=0

IFS=$'\n' # Esto es para q el for separe por newline y no por blankspace

for ARCH\_PATH in $(find "$DATADIR" -type f) ; do

# Nos quedamos con el nombre del archivo (descartamos el path)

ARCH=$(basename "$ARCH\_PATH")

# Validamos q el formato del nombre sea correcto:

if [ -z $(echo "$ARCH" | grep '^vta.\*\.txt$') ] ; then

echo "$ARCH formato de archivo invalido"

continue

fi

# Nos quedamos con el codigo de Sucursal:

SUC=$(echo "$ARCH" | sed 's:^vta\(.\*\)\.txt$:\1:g')

# Validamos que la sucursal exista y este habilitada:

if [ -z "$(cat "$ARCH\_SUC" | grep "^${SUC};[^;]\*;S$")" ] ; then

echo "Sucursal $SUC inexistente o inhabilitada."

continue

fi

# Procesamos la data del archivo para nuestro cliente:

# Filtramos los registros del cliente (con el grep)

# Y para c/u emitimos <cantidad>\*<precio> al array MONTO\_ARCH (con el sed)

# O sea, expresamos la cuenta del Monto Parcial...luego abajo calculamos el resultado

MONTO\_ARCH=$(cat "$ARCH\_PATH" | grep "^[^;]\*;[^;]\*;${COD\_CLI};[^;]\*;[^;]\*$" | sed "s:^.\*;\([^;]\*\);\([^;]\*\)$:\1\\*\2:g")

# Si el array no tiene elementos, seguimos con otro archivo:

if [ -z "$MONTO\_ARCH" ] ; then

continue

fi

# Procesamos los parciales (hacemos la cuenta) y acumulamos en el Total

IFS=$'\n' # Esto es para q el for separe por newline y no por blankspace

for LI in $(echo "$MONTO\_ARCH") ; do

MONTO\_PARCIAL=$(($LI))

MONTO\_TOTAL=$(($MONTO\_TOTAL+$MONTO\_PARCIAL))

done

done # for

# FIN! Resultado final

echo "MONTO TOTAL FINAL PARA CLIENTE $COD\_CLI: $MONTO\_TOTAL"

exit