**Cátedra:** Osvaldo Clua  
**Fecha:** Primera Oportunidad - Segundo Cuatrimestre 2006  
**Día:** 22/10/2006

**Enunciado**

**Expresiones Regulares**

Se requiere desarrollar un script parser.sh tal que dado un archivo fuente pasado por parámetro, genere un archivo de salida con el mismo contenido del archivo de entrada, excepto los caracteres espacio y tabs que se encuentren en las lineas, además no debe aparecer en el archivo de salida ninguna linea sin caracter.

El archivo de salida debe generarse en el directorio /tmp, con nombre igual al del archivo original, mas el agregado de una extensión ”-+-”

**PERL**

Dados N archivos identificados por la máscara “accesos.txt” residentes en el directorio actual de trabajo, que poseen información de la cantidad de accesos a cada página de un sitio web, por día de la semana para un mes determinado con el siguiente formato:

accesos.txt

martes;pag1;1546

martes;pag45;25

lunes;pag10;6476

jueves;pag1;658

lunes;pag3;77

Notas:

1. Puede haber meses para los cuales ninguna pagina haya tenido accesos
2. Puede haber paginas para las cuales no hayan habido accesos en algun dia
3. Pueden venir por error archivos vacios
4. En distintos archivos puede venir informacion correspondiente al mismo día y página, que se debera tomar como válida.

Se pide programar un comando en Perl que emita un listado totalizando la cantidad de accesos a cualquier pagina por día de la semana. En el supuesto que para un día no haya habido accesos a ninguna página, en el listado, en vez del total de accesos para ese día, debera figurar la leyenda “NO HUBO ACCESOS”. El listado deberá estar ordenado por orden cronológico de los días de la semana comenzando el domingo.

Ejemplo de listado:

lunes 1678

martes 977

miercoles NO HUBO ACCESOS

**BASH**

Una entidad bancaria registra mensualmente información sobre los créditos otorgados a sus clientes, la cual se almacena con el siguiente formato:

Nombre de archivo: Prestamos.<aaaamm>.txt

Estructura:

* Código de Cliente: Caracteres
* Dia: caracteres
* Linea crédito: Caracteres
* Importe solicitado: Caracteres

Ordenado ascendentemente por día y linea de credito. Ej: 7508;18;A21;10000

Se cuenta, además con un archivo donde se registran las lineas de credito que la entidad bancaria maneja, con el formato:

Nombre de archivo: Lineas.txt

Estructura:

* Linea de Crédito: carateres
* Tasa vigente: Caracteres (nn,nn)
* Plazo: Caracteres
* Importe máximo: caracteres

Ordenado ascendentemente por linea de crédito.

Se pide:

Desarrollar un comando que reciba como parametros obligatorios: uno o mas meses(aaaamm). Este comando debera generar un archivo que contenga la informacion de los prestamos discriminada por linea de crédito, cliente, para los meses seleccionados, con el siguiente formato:

Nombre de archivo: informe.txt

Estructura:

* Linea de crédito: Caracteres
* Código de cliente: caracteres
* intereses(\*): caracteres
* importe maáximo: caracteres

(\*) intereses = importe solicitado \* tasa vigente \* plazo

Debe ademas imprimir por salida estandar un total de archivos procesados, con la cantidad total de registros. Si para alguno de los meses pasados por parametro no existiera archivo indicarlo tambien con un mensaje en esta salida.

El archivo debe estar ordenado ascendentemente por linea de credito y cliente.

NOTA: en todos los archivos el separador de campos es ”;”.

**Resolución**

**Expresiones Regulares**

if [ $# -ne 1 ]

then

echo "Debe indicar un archivo. Uso: `basename $0` file"

exit 1

fi

INPUT=$1

OUTPUT="tmp/$INPUT.+-+"

# Saco todos los tabs y espacios en blanco y a eso le saco todas las lineas vacias

sed -e "s/[ \t]//g" -e "/^$/d" "$INPUT" > "$OUTPUT"

**PERL**

# Idea: Hago un hash por día, al que le voy sumando elementos

my $FILES\_PATTERN="accesos\*.txt";

my $NO\_HUBO\_ACCESOS="NO HUBO ACCESOS";

my @dias=("domingo", "lunes", "martes", "miercoles", "jueves", "viernes", "sabado");

my %hashDeAccesos;

# Retorna un vector con los archivos que matchean la regexp de accesos

sub obtenerArchivosDeAccesos {

my $lsresult= `ls $FILES\_PATTERN 2> /dev/null`;

my @archivos=[split](http://perldoc.perl.org/functions/split.html)(" ",$lsresult);

[return](http://perldoc.perl.org/functions/return.html) @archivos;

}

# Agrega los accesos a la pagina para cada dia

sub procesarArchivo {

my ($archivo)=@\_;

[print](http://perldoc.perl.org/functions/print.html) "Voy a procesar a $archivo\n";

[open](http://perldoc.perl.org/functions/open.html) (fdArchivo,$archivo);

while (<fdArchivo>){

[chomp](http://perldoc.perl.org/functions/chomp.html);

$linea=$\_;

my ($dia,$pagina,$accesos)=[split](http://perldoc.perl.org/functions/split.html)(/\;/,$linea);

$hashDeAccesos{$dia}+=$accesos;

}

[close](http://perldoc.perl.org/functions/close.html) (fdArchivo);

}

my @archivos=obtenerArchivosDeAccesos();

foreach $archivo (@archivos) {

procesarArchivo("$archivo");

}

foreach $dia (@dias) {

my $accesos = $NO\_HUBO\_ACCESOS;

if ([exists](http://perldoc.perl.org/functions/exists.html) $hashDeAccesos{$dia}) {

$accesos = $hashDeAccesos{$dia}

}

[print](http://perldoc.perl.org/functions/print.html) "$dia $accesos\n";

}

**BASH**

# Constantes usadas

ARCHIVO\_LINEAS="Lineas.txt"

SEPARADOR=";"

ARCHIVO\_SALIDA="informe.txt"

# Contadores usados

CANTIDAD\_ARCHIVOS=0

CANTIDAD\_REGISTROS=0

# Uso: split RECORD SEPARATOR

# Deja en VALUES, todos los campos del registro

# Retorna la cantidad de campos encontrados en el registro

function split(){

unset VALUES

VALUES=( `echo "$1" | tr -s "'$2'" ' '` )

return ${#VALUES[\*]}

}

# Uso: cargarInfoLinea codigoLinea

# Deja en INFOLINEA: Linea de Credito, Tasa vigente, Plazo, Importe maximo

# Retorna 0 si encuentra la linea de credito en $ARCHIVO\_LINEAS o 1 si no

function cargarInfoLinea() {

unset INFOLINEA

local linea=`grep "^$1" "$ARCHIVO\_LINEAS"`

if ! [ -z $linea ]

then

split $linea $SEPARADOR

if [ $? -eq 4 ]

then

INFOLINEA=(${VALUES[@]})

return 0

fi

fi

return 1

}

# Uso: agregarPedidoAInforme pedido

# Agrega a $ARCHIVO\_SALIDA las informaciones sobre el pedido particular

function agregarPedidoAInforme() {

split "$1" $SEPARADOR

#Codigo de Cliente;Dia;Linea credito;Importe solicitado

local linea=${VALUES[2]}

local cliente=${VALUES[0]}

local importe=${VALUES[3]}

cargarInfoLinea $linea

if [ $? -eq 0 ]

then

local importeMaximo=${INFOLINEA[3]}

local tasa=${INFOLINEA[1]}

local plazo=${INFOLINEA[2]}

local intereses=`echo "$importe\*$tasa\*$plazo" | bc`

local lineaInforme="$linea$SEPARADOR$cliente$SEPARADOR$intereses$SEPARADOR$importeMaximo"

echo "$lineaInforme" >> $ARCHIVO\_SALIDA

let CANTIDAD\_REGISTROS+=1

else

echo "El cliente $cliente pidio $importe en la linea $linea, pero la linea no existe"

fi

}

# Uso: agregarPeriodoAInforme periodo

# Agrega a $ARCHIVO\_SALIDA las informaciones sobre creditos a otorgar del periodo

# Retorna 0 si logra agregar informacion, o 1 si no existe el archivo Prestamos.periodo.txt

function agregarPeriodoAInforme() {

if [ -f "Prestamos.$1.txt" ]

then

while read pedido

do

agregarPedidoAInforme "$pedido"

done < "Prestamos.$1.txt"

let CANTIDAD\_ARCHIVOS+=1

return 0

fi

return 1

}

function ordenarInforme() {

sort -t$SEPARADOR -k1 -k2 $ARCHIVO\_SALIDA >> $ARCHIVO\_SALIDA.tmp #las opciones no son necesarias, ya que alfabeticamente da igual

mv $ARCHIVO\_SALIDA.tmp $ARCHIVO\_SALIDA

}

if [ $# -lt 1 ]

then

echo "Debe indicar al menos un periodo"

exit 1

fi

for periodo in $@

do

agregarPeriodoAInforme $periodo

if [ $? -ne 0 ]

then

echo "No existe el archivo Prestamos.$periodo.txt"

fi

done

echo "Se procesaron $CANTIDAD\_ARCHIVOS archivos con un total de $CANTIDAD\_REGISTROS registros validos"

ordenarInforme

exit 0