**Cátedra:** Osvaldo Clua  
**Fecha:** Primera Oportunidad - Primer Cuatrimestre 2004  
**Día:** 01/06/2004

**Enunciado**

**Expresiones Regulares**

Un organismo previsional necesita que desarrollemos una solución rápida que responda a la siguiente pregunta:  
**¿Cuántas jubilaciones se deberán pagar el próximo mes?**

La ley que rige para este organismo dice que se pagará una jubilación a todo aportante que este en tramite de jubilarse y que tenga por lo menos 200 aportes. Se encuentra en tramite de jubilarse todo aportante activo, con Tipo de Tramite “JUILACION” y Fecha Fin de Tramite en nulo. La información se encuentra en un archivo llamado **“TramitesAportantes“**, con los siguientes campos de longitud variable y separados por el caracter ”]”:

* TIPO\_DE\_DECUMENTO\_APORTANTE
* NUMERO\_DE\_DOCUMENTO\_APORTANTE
* TIPO\_DE\_APORTANTE
* CANTIDAD\_DE\_APORTES
* NOMBRE\_DE\_APORTANTE
* ESTADO\_CIVIL
* ACTIVO
* INCAPACIDAD
* TIPO\_DE\_DOCUMENTO\_BENEFICIARIO
* NUMERO\_DE\_DOCUMENTO\_BENEFICIARIO
* TIPO\_TRAMITE
* FECHA\_INICIO
* FECHA\_FIN\_TRAMITE

***IMPORTANTE:* Utilizar solamente comandos que acepten expresiones regulares, wc y/o cat.**

**PERL**

Se tiene una máquina servidora, en la que se requiere disponer de un ejecutor de comandos, al que llamaremos TASKS, que le permita a los operadores ejecutar determinadas tareas en base a archivos residentes en el directorio que le pasarán como parámetro. TASKS deberá procesar todos los archivos de dicho directorio y ejecutar un comando por cada uno de ellos, según indique el archivo de configuración **sever.config**, residente en el mismo directorio que TASKS. Cada línea de este archivo tiene alguno de los siguientes formatos: (el separador tanto para FILE como para CMD es ”:”)

* **CMD <cant. total de parámetros><cmd>**.

Ejemplo: CMD:8:teleproc

Este tipo de línea indica que el comando **<cmd>** acepta la cantidad de parámetros indicada en **<cant total de parámetros>**: Esta cant total de paráms debe ser la suma de los 'fijos' mas los 'variables': Chequear que sea asi (es decir que en el archivo venga la cantidad que corresponde).

* **FILE <cmd><nombre archivo><lista parámetro fijos separados por '|'>**

Ejemplo: FILE:teleproc:telefonos.txt:-f|-o \tmp|-r no-hung

Este tipo de linea indica que para procesar un archivo con nombre **<nombre archivo>** se deberá ejecutar el comando **<cmd>**, con ciertos parámetros: una cantidad de parámetro (fijos) indicada en **<lista de parametros fijos separados por '|'>** seguida de una cantidad de parámetros variables que constituyen la primera linea del archivo siendo procesado y tambien están separados por '|'.

Se pide escribir en PERL, el script TASKS.pl que recibiendo como parámetro el directorio donde se encuentran los archivos a procesar, primero cargue en estructuras de memoria adecuadas la información de configuración, y luego afectúe el procesamiento de los arvhiso. Como resultado, deberá mostrar por pantalla el porcentaje de comando para lso que no coincidió la cantidad total de parámetros especificada, el porcentaje otros errores, y el porcentaje de ejecutados exitosamente.

*Nota:* resolver haciendo uso de hashes, arreglos y de la función split, siempre que resulte adecuado y simplifique el desarrollo

**BASH**

Se tiene un archivo con parámetros de usuarios, llamado **PARAMETROS.DAT**, con el siguiente formato:  
**usuario;directorio origen;grupo;cant.max de archivos**

Se pide hacer un script que proces el archivo de la sgte manera:

1. Si el usuario que ejecuta el script no existe en el archivo de parámetros, cancelarlo con el siguiente mensaje: <usuario que ejecuta el script> **–usuario inexistente**
2. Luego por cada registro, efectúe las siguientes validaciones y/u operaciones y grabe lo indicado en el archivos de log LOG.TXT:
   1. Si el direcotorio origen del usuario no existe, grabar: **<usuario> sin directorio origen**
   2. Si la cantidad de archivos (comunes) dentro del directorio origen es mayor que el máximo, grabar: **<usuario> supera el límite de archivos**
   3. Calcular el total de **programas PERL** que hay en el directorio origen (\*) y grabar en el log: **<usuario> tiene nn programas PERL**

(\*) Para determinar el tipo de archivo analizar la primera linea del mimso

**Resolución**

**Expresiones Regulares**

#!/bin/bash

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

# Parcial 20040601\_1 : Ejercicio REGEX

# Resolucion

# Autor: Maximiliano Milicich (mmilicich)

#

# HIPOTESIS Y PRECONDICIONES:

#

# - El ultimo campo de un registro NO lleva separador de campo final

# - <Fecha Fin de Tramite> NULO significa que NO HAY DATOS PARA ESE CAMPO

# - El campo activo posee unicamente los valores [SI-NO]. SI = activo. NO = inactivo

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

JUB\_SEP=']' # separador de campos

JUB\_NOSEP="[^${JUB\_SEP}]" # cualquier caracter que no sea un separador de campos

JUB\_ACTIVO='SI' # Indicador de registro ACTIVO

JUB\_TIPO\_TRAMITE\_JUBILACION='JUBILACION' # Tipo de Tramite JUBILACION

JUB\_DATAFILE='TramitesAportantes' # Archivo de datos a procesar

JUB\_TEMPFILE='/tmp/jub\_count.tmp' # Archivo temporal auxiliar

# Expresion para el GREP: filtra los registros que cumplen las condiciones pedidas

# notar que ${JUB\_CANT\_APORTES} es variable, por eso la necesidad de una funcion que "reevalue" la expresion

# Para hacer mas clara la construccion de la REGEX, la separamos en pasos:

function set\_jub\_grep\_expr()

{

# Los 3 primeros campos pueden ser cualquier cosa:

JUB\_GREP\_EXPR="^\(${JUB\_NOSEP}\*${JUB\_SEP}\)\{3\}"

# Luego viene el filtro para el campo Cantidad de aportes:

JUB\_GREP\_EXPR=${JUB\_GREP\_EXPR}"${JUB\_CANT\_APORTES}${JUB\_SEP}"

# Luego, 2 campos mas que pueden ser cualquier cosa:

JUB\_GREP\_EXPR=${JUB\_GREP\_EXPR}"\(${JUB\_NOSEP}\*${JUB\_SEP}\)\{2\}"

# Luego viene el filtro por ACTIVOS:

JUB\_GREP\_EXPR=${JUB\_GREP\_EXPR}"${JUB\_ACTIVO}${JUB\_SEP}"

# Luego, 3 campos mas que pueden ser cualquier cosa:

JUB\_GREP\_EXPR=${JUB\_GREP\_EXPR}"\(${JUB\_NOSEP}\*${JUB\_SEP}\)\{3\}"

# Luego el filtro por Tipo de tramite (JUBILACION):

JUB\_GREP\_EXPR=${JUB\_GREP\_EXPR}"${JUB\_TIPO\_TRAMITE\_JUBILACION}${JUB\_SEP}"

# Por ultimo, el campo de Fecha\_Inicio (puede ser cualquier cosa)

# Y luego de este, viene un separador y el final de linea (caracter de anclaje $):

# Esto es senial de que el campo fecha\_fin\_tramite es NULO

JUB\_GREP\_EXPR=${JUB\_GREP\_EXPR}"${JUB\_NOSEP}\*${JUB\_SEP}$"

} # end function set\_jub\_grep\_expr()

# filtramos primero los que tienen cantidad de 200 a 999 (pueden haber ceros a izquierda)

JUB\_CANT\_APORTES='0\*[2-9][0-9]\{2\}'

set\_jub\_grep\_expr

grep "$JUB\_GREP\_EXPR" "$JUB\_DATAFILE" > "$JUB\_TEMPFILE"

# luego filtramos los que tienen cantidad de 1000 en adelante (pueden haber ceros a izquierda)

JUB\_CANT\_APORTES='0\*[1-9][0-9]\{3,\}'

set\_jub\_grep\_expr

grep "$JUB\_GREP\_EXPR" "$JUB\_DATAFILE" >> "$JUB\_TEMPFILE"

# Por ultimo, contamos los registros filtrados (SOLUCION) y nos vamos:

cat "$JUB\_TEMPFILE" | wc -l

rm "$JUB\_TEMPFILE"

exit

**BASH**

#!/bin/bash

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

# FI.UBA.AR 75.08

# Parcial 20040601\_1 - Ejercicio BASH

# Resolucion

# Autor: Maximiliano Milicich (mmilicich)

#

# HIPOTESIS Y PRECONDICIONES:

#

# - FALTA HACER EL PUNTO 2.3 (el de programas PERL) !!!!!!!

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

USU\_PARMFILE='PARAMETROS.DAT'

USU\_LOGFILE='LOG.TXT'

USU\_THISUSER=`whoami` # el usuario que ejecuta el script

# Validamos si el usuario existe:

USU\_VALIDUSER=$(grep "^$USU\_THISUSER;.\*$" "$USU\_PARMFILE")

if [ -z "$USU\_VALIDUSER" ]

then

echo "$USU\_THISUSER - usuario inexistente"

exit 1

fi

# Recorremos el archivo de usuarios realizando las validaciones pedidas:

while read USU\_LINEA

do

if [ -z "$USU\_LINEA" ] ; then

continue

fi

USU\_USER=$(echo "$USU\_LINEA" | cut -f1 -d';')

USU\_SDIR=$(echo "$USU\_LINEA" | cut -f2 -d';')

USU\_GRUP=$(echo "$USU\_LINEA" | cut -f3 -d';')

USU\_MAXF=$(echo "$USU\_LINEA" | cut -f4 -d';')

if [ ! -d "$USU\_SDIR" ]

then

echo "$USU\_USER sin directorio origen" >> "$USU\_LOGFILE"

continue

fi

if [ $(find "$USU\_SDIR" -type f 2> /dev/null | wc -l) -gt $USU\_MAXF ]

then

echo "$USU\_USER supera el limite de archivos" >> "$USU\_LOGFILE"

continue

fi

done < "$USU\_PARMFILE"

exit