CARÁTULA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tema** | **Grupo** | **Ayudante** |  | **Evaluación Grupal del TP** | | |
|  | 01 |  | **Fecha** | **09/05/2013** | | |
| **Hora Inicio** |  | **Hora Fin** |  |
| **Firma:** | **Nota** |  | | |

INTEGRANTES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Padrón** | **Apellido y Nombre** | **Asistencia a Entrega** | **Asistencia a Revisión** | **Evaluación Individual Final** |
| 1 | 88896 | Bayetto, Ignacio Javier |  |  |  |
| 2 | 90009 | Medrano, Lautaro |  |  |  |
| 3 | 90697 | Eisner, Ariel |  |  |  |
| 4 | 91525 | De Zan, Ignacio |  |  |  |
| 5 | 91545 | Disalvo, Francisco Andrés |  |  |  |
| 6 | 93606 | Finkelstein, Damián |  |  |  |

PLANILLAS DE EVALUACIÓN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **DOCUMENTACIÓN** |  |  |  |  |
| Carpeta: sin hojas sueltas? |  |  |  |  |
| Carátula: 2 copias y completa? |  |  |  |  |
| Índice y pie de página completos? |  |  |  |  |
| Documenta hipótesis y problemas relevantes? |  |  |  |  |
| Documenta el Readme correctamente? |  |  |  |  |
| Indentación del código y con comentarios? |  |  |  |  |
| Documenta las estructuras y los casos de prueba? |  |  |  |  |
| Documenta la ejecución de pruebas y muestra la SALIDA? |  |  |  |  |
| Incluye este  enunciado? |  |  |  |  |
| Trae el TP en un dispositivo listo? |  |  |  |  |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

CARÁTULA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tema** | **Grupo** | **Ayudante** |  | **Evaluación Grupal del TP** | | |
|  | 01 |  | **Fecha** | **09/05/2013** | | |
| **Hora Inicio** |  | **Hora Fin** |  |
| **Firma:** | **Nota** |  | | |

INTEGRANTES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Padrón** | **Apellido y Nombre** | **Asistencia a Entrega** | **Asistencia a Revisión** | **Evaluación Individual Final** |
| 1 | 88896 | Bayetto, Ignacio Javier |  |  |  |
| 2 | 90009 | Medrano, Lautaro |  |  |  |
| 3 | 90697 | Eisner, Ariel |  |  |  |
| 4 | 91525 | De Zan, Ignacio |  |  |  |
| 5 | 91545 | Disalvo, Francisco Andrés |  |  |  |
| 6 | 93606 | Finkelstein, Damián |  |  |  |

PLANILLAS DE EVALUACIÓN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **DOCUMENTACIÓN** |  |  |  |  |
| Carpeta: sin hojas sueltas? |  |  |  |  |
| Carátula: 2 copias y completa? |  |  |  |  |
| Índice y pie de página completos? |  |  |  |  |
| Documenta hipótesis y problemas relevantes? |  |  |  |  |
| Documenta el Readme correctamente? |  |  |  |  |
| Indentación del código y con comentarios? |  |  |  |  |
| Documenta las estructuras y los casos de prueba? |  |  |  |  |
| Documenta la ejecución de pruebas y muestra la SALIDA? |  |  |  |  |
| Incluye este  enunciado? |  |  |  |  |
| Trae el TP en un dispositivo listo? |  |  |  |  |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **DESARROLLO y PRUEBA UNITARIA** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **PRUEBA** |  |  |  |  |
| Capacidad para la aplicación de cambios |  |  |  |  |
| Test con Datos sin errores |  |  |  |  |
| Test con Datos con errores |  |  |  |  |
| Manejo de  errores |  |  |  |  |
| Logueo |  |  |  |  |
| Encadenamiento de comandos |  |  |  |  |
| Funcionamiento General del TP |  |  |  |  |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **INTEGRACION** |  |  |  |  |
| Encadenamiento de comandos |  |  |  |  |
| Funcionamiento General del TP |  |  |  |  |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **DESARROLLO y PRUEBA UNITARIA** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **PRUEBA** |  |  |  |  |
| Capacidad para la aplicación de cambios |  |  |  |  |
| Test con Datos sin errores |  |  |  |  |
| Test con Datos con errores |  |  |  |  |
| Manejo de  errores |  |  |  |  |
| Logueo |  |  |  |  |
| Encadenamiento de comandos |  |  |  |  |
| Funcionamiento General del TP |  |  |  |  |
| **Ítem** | **Aprobado** | **A Revisión** | **No posee** | **Responsable, Observaciones** |
| **INTEGRACION** |  |  |  |  |
| Encadenamiento de comandos |  |  |  |  |
| Funcionamiento General del TP |  |  |  |  |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |
|  | | | | |

**Índice**

1. **Hipótesis y aclaraciones globales 6**
2. **Problemas relevantes 7**
3. **Archivo README 9**
4. **Listado de comandos y funciones**

**4.1 - InstalarX.sh 10**

**4.2 - InicioX.sh 30**

**4.3 - DetectarX.sh 35**

**4.3.1 - valPais.sh 39**

**4.4 - Interprete.sh 41**

**4.5 - ReporteX.pl 48**

**4.6 - GlogX.sh 67**

**4.7 - VlogX.sh 69**

**4.8 - StartX.sh 71**

**4.9 - StopX.sh 73**

**4.10 - MoverX.sh 74**

1. **Archivos 76**
2. **Directorios, Datos y Salida 77**
3. **Enunciado 79**

1 - Hipótesis y aclaraciones globales

Hemos decidido presentar las hipótesis por cada comando y no en un único capítulo. Creemos que esto ayuda a clarificar la documentación.

**2 - Problemas relevantes**

En esta sección mencionaremos los problemas encontrados durante el desarrollo y las soluciones que se aplicaron para resolverlos.

* Al probar el trabajo práctico en las computadoras del laboratorio de la facultad, encontramos que los scripts no se comportaban de igual forma que en nuestras computadoras personales, por lo que tuvimos que hacer varios cambios para lograr un funcionamiento estándar.
* En InstalarX.sh tuvimos problemas a la hora de encontrar la forma para validar que Perl estuviera instalado en el ambiente. Lo solucionamos mediante la siguiente función:

|  |
| --- |
| #Valida si el usuario tiene instalado perl. De ser asi muestra la version del mismo. En caso contrario muestra |
| #un mensaje y sale de la instalacion. |
| function validarPerl { |
|  |
| pathPerl=`which perl` |
| validacionPerl="false" |
|  |
| if [ "$?" == 0 ]; then |
| #6.2 - Perl esta instalado. Mostrar y grabar en el log un mensaje informativo con la version de Perl que se encuentra instalada. |
| version=` $pathPerl -v | grep "v[5-9]\."` |
| if [ -n "$version" ]; then |
| validacionPerl="true" |
| mensaje=" |
| TP SO7508 1er cuatrimestre 2013. Tema T Copyright (c) Grupo 01. |
| Perl Version: $version |
| " |
| echo "$mensaje" |
|  |
| grabarLog "I" "$mensaje" |
|  |
| fi |
| fi |
|  |
| if [ $validacionPerl == "false" ]; then |
| #6.1 - Validacion de Perl da error. Mostrar y grabar en el log. |
| mensaje=" |
| TP S07508 1er cuatrimestre 2013. Tema T Copyright (C) Grupo 01. |
| Para instalar el TP es necesario contar con  Perl 5 o superior instalado. Efectúe su instalación e inténtelo nuevamente. |
|  |
| Proceso de Instalación Cancelado. |
| " |
| echo "$mensaje" |
|  |
| grabarLog "I" "$mensaje" |
|  |
| exit 0 |
| Fi |
|  |
| } |

* En InicioX.sh a la hora de la ejecucion, notamos que si lo ejecutábamos con “./”los cambios en las variables de entorno no se verian reflejados una vez terminado el script. Por lo cual optamos por ejecutarlo de la siguiente manera: ". InicioX.sh"(punto espacio).
* En Interprete.sh tuvimos problemas a la hora de recorrer los archivos que se encontraban en el directorio de aceptados. Notamos que estábamos perdiendo una línea al salir de un while. Lo solucionamos ejecutando el while de la siguiente manera:

|  |
| --- |
| while IFS=$'\n' read -r linea || [[ -n "$linea" ]] |
| do |
| […]  done < $ACEPDIR/$archivo |

* En StartX.sh tuvimos problemas para validar que un proceso no se esté ejecutando antes de lanzarlo con la función StartX.sh. Al buscar si ya había un proceso corriendo con el nombre del comando a ejecutar mediante:

|  |
| --- |
| var=`ps -fea | grep -v "grep" | grep -v "StartX" | grep "$nombreComando" | wc -l` |
| if [ "$var" -ne 0 ] |
| then |
| GlogX "$comando.sh" "SE" "StartX.sh encuentra un proceso $nombreComando en  ejecución" "$comando" |
| return 1 |
| Fi |
|  |

La computadora del laboratorio C de la facultad, nos detectaba un proceso ejecutándose independientemente del nombre del comando que le pasáramos. Luego de un tiempo investigando, nos dimos cuenta que la causa de este problema era tener abierto el Gedit para editar los .sh

* En DetectaX.sh notamos que al correr el demonio en background y que el mismo devolviera una salida por consola, nos sacaba el control de la misma por lo que al estabilizar dicho script optamos por no mostrar nada por consola y pasar todo al DetectaX.log.

3 – Archivo README

Para proceder a realizar una correcta instalación deberán tenerse en cuenta los siguientes pasos previos establecidos en el archivo README.

Los mismos se muestran a continuación:

README GRUPO 01

1- Abrir una terminal, posicionarse donde se encuentre el archivo grupo01.tar.gz y extraer el contenido del mismo ejecutando el comando tar xvzf grupo01.tar.gz.

2- Moverse hasta el directorio grupo01/instalacion con el comando: cd grupo01/instalacion

3- Luego, parado desde grupo01/instalacion, ejecutar el comando: ./InstalarX.sh

4- Una vez terminado el proceso de instalación, volvemos al directorio inicial (grupo01), nos paramos en el directorio de binarios (cuyo nombre elegimos en el proceso de instalación), aquí se asume que fue bin e inicializamos el ambiente de la siguiente manera: cd ../bin/. InicioX.sh

5- Una vez iniciado el ambiente, el demonio ya estará corriendo preparado para recibir los archivos externos en el directorio de arribos y proceder con la correcta ejecución del programa.

6- En caso de querer detener la ejecución del demonio, ejecutamos un ps en la consola obtenemos el número de proceso del demonio y ubicados en el directorio de binarios de grupo01, ejecutamos el siguiente comando: ./StopX.sh NNNNNN donde NNNNNN es el número de proceso.

7- En caso de querer reanudar la ejecución del demonio, ubicados en el directorio de binarios de grupo01, ejecutamos el siguiente comando: ./StartX.sh "<comando invocante>" "DetectaX.sh <Nro ciclos> <Tiempo de espera>"

Cabe aclarar que al realizar el paquete de instalación dimos por supuestas las siguientes cuestiones:

* Se agrupó todo el paquete en un directorio llamado instalacion, el cual contiene el comando InstalarX.sh y tres directorios, archivos ejecutables (bin), archivos maestros (mae) y tablas de configuración (tablas), los que se necesitan para poder ejecutar el programa.
* El directorio instalacion previamente mencionado se encuentra comprimido, junto con el archivo README, en grupo01.tar.gz.

4 – Listado de Comandos y Funciones

### **4.1 - InstalarX.sh**

**Título:** InstalarX.sh

**Justificación**

Este comando es el encargado de realizar la instalación de CONTROLX. Creará los directorios necesarios para el correcto funcionamiento del sistema interactuando con el usuario y realizando todas las validaciones que sean necesarias. Además es el encargado de crear el archivo de configuración del sistema InstalarX.conf y el archivo de log de instalación InstalarX.log.

**Archivos de Input, Intermedios y de Output**

Archivos Input:

InstalarX.sh toma los archivos ejecutables de $GRUPO/instalacion/bin y los mueve al directorio para ejecutables definido por el usuario durante la instalación.

Los archivos ejecutables son:

* InicioX.sh
* DetectaX.sh
* valPais.sh
* Interprete.sh
* ReporteX.sh
* GlogX.sh
* VlogX.sh
* MoverX.sh
* StartX.sh
* StopX.sh

InstalarX.sh toma los archivos maestros de $GRUPO/instalacion/mae y los mueve al directorio de maestros definido por el usuario durante la instalación.

Los archivos maestros son:

* ps-mae.mae
* PPI.mae

InstalarX.sh toma las tablas de configuración de $GRUPO/instalación/tablas y las mueve al directorio de configuración de CONTROLX.

Las tablas de configuración son:

* T1.tab
* T2.tab

Archivos Output:

InstalarX.sh crea el log de instalación en $CONFDIR: InstalarX.log

InstalarX.sh crea el archivo de configuración de CONTROLX en $CONFDIR: InstalarX.conf

**Parámetros y Opciones**

No tiene parámetros.

**Invocación Manual y/o automática con ejemplos**

Este comando debe ser invocado manualmente desde la consola de la siguiente manera:

./InstalarX.sh

**Hipótesis y Aclaraciones especificas del comando**

Pasaremos a plantear los supuestos realizados, los problemas encontrados a la hora de realizar éste comando y pruebas del uso del mismo.

Definición de directorios

* Para evitar confusiones a la hora de ingresar los nombres de los directorios, en vez de mostrar la ruta completa, se sugiere el nombre del directorio teniendo el usuario la posibilidad de cambiarlo agregando uno o más directorios a partir de $GRUPO. O sea que se considera el path relativo y no el absoluto.
* Por ejemplo, para el directorio de arribos se muestra como sugerencia arribos, quedando el path completo como $GRUPO/arribos. Si se desea agregar para los arribos un directorio con subdirectorios quedando $GRUPO/archivos/arribos, el usuario deberá ingresar archivos/arribos. Sucede lo mismo para todos los demás directorios que puede definir el usuario.
* En caso de que el usuario no ingrese nada no sea directamente presione enter, se tomará por defecto el nombre de directorio establecido en la sugerencia.

Validación de campos numéricos

* Cuando al usuario se le solicita ingresar el espacio máximo para el arribo de archivos externos, se valida que el número ingresado sea un número entero y no decimal y que además no pueda ingresar 0 como opción.
* Otro de los campos numéricos que se define es el tamaño máximo de los archivos de log, para el cual se dan por supuestas las mismas validaciones que para el espacio máximo de archivos externos.
* En ambos casos, si el usuario no ingresa nada y presiona enter, se toma el valor definido en la sugerencia.

Validación de existencia de archivos binarios y maestros

* Al comenzar la instalación, uno de las primeras validaciones que se realizan es que existan en el directorio instalacion/bin todos los archivos binarios necesarios para la correcta ejecución del programa.
* Lo mismo sucede con los archivos maestros, para realizar una correcta instalación se debe contar con todos los archivos maestros en el directorio instalación/mae.

Uso del comando GlogX

* Este comando requiere de la creación de un archivo de log en el directorio CONFDIR. Como en la instalación no se cuenta con los directorios definidos por el usuario se usa el comando GlogX directamente desde el directorio de ejecutables propio de la instalación (instalacion/bin).

Uso del comando MoverX

* Este comando requiere poder mover archivos de un directorio a otro por lo que recurre a utilizar el comando MoverX. Como en la instalación no se cuenta con los directorios definidos por el usuario se usa el comando MoverX directamente desde el directorio de ejecutables propio de la instalación (instalación/bin).

Interacción con el usuario

* En aquellos casos en que el programa solicite el ingreso de una respuesta como [si/no], solamente se da como válida la respuesta cuando el usuario ingresa 's' o 'n', en todos los demás casos se vuelve a realizar la pregunta.

**Código fuente**

|  |
| --- |
| #!/bin/bash |
|  |
| #=========================================================== |
| # ARCHIVO: InstalarX.sh |
| # |
| # FIRMA: ./InstalarX.sh |
| # |
| # DESCRIPCION: Comando utilizado para realizar la instalacion de ControlX |
| # |
| # AUTOR: Bayetto, Ignacio Javier. |
| # PADRON: 88896 |
| # |
| #=========================================================== |
|  |
|  |
| #Lee las variables del archivo de configuración. |
| function leerVariablesDeConfiguracion { |
|  |
| GRUPO=`grep "GRUPO" "$1" | cut -d"=" -f 2` |
| BINDIR=`grep "BINDIR" "$1" | cut -d"=" -f 2` |
| MAEDIR=`grep "MAEDIR" "$1" | cut -d"=" -f 2` |
| ARRIDIR=`grep "ARRIDIR" "$1" | cut -d"=" -f 2` |
| ACEPDIR=`grep "ACEPDIR" "$1" | cut -d"=" -f 2` |
| RECHDIR=`grep "RECHDIR" "$1" | cut -d"=" -f 2` |
| PROCDIR=`grep "PROCDIR" "$1" | cut -d"=" -f 2` |
| REPODIR=`grep "REPODIR" "$1" | cut -d"=" -f 2` |
| LOGDIR=`grep "LOGDIR" "$1" | cut -d"=" -f 2` |
| LOGEXT=`grep "LOGEXT" "$1" | cut -d"=" -f 2` |
| LOGSIZE=`grep "LOGSIZE" "$1" | cut -d"=" -f 2` |
| DATASIZE=`grep "DATASIZE" "$1" | cut -d"=" -f 2` |
|  |
| } |
|  |
| #Valida la extencion del archivo de log |
| function validarExtensionLog { |
|  |
| res=`echo "$1" | grep "^[.]"` |
| if [  "$1" != "$res" ]; then |
| echo "ERROR - La extensión del log debe comenzar con un punto (.)" |
| grabarLog "E" "La extensión del log debe comenzar con un punto (.)" |
| cond="error" |
| fi |
| unset res |
| } |
|  |
| #Define el tamanio maximo para los archivos de log |
| function definirLogSize { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| #18 - Definir el tamanio maximo para los archivos de log |
| echo " |
| \* Defina el tamaño máximo para los archivos de Log (en Kbytes): |
| \* Sugerencia: ($LOGSIZE)" |
| read maxSize |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $maxSize ];then |
| maxSize=$LOGSIZE |
| fi |
|  |
| validarNumerico $maxSize |
| done |
|  |
| LOGSIZE=$maxSize |
| unset cond |
|  |
| } |
|  |
| #Define la extensión de los archivos de log |
| function definirLogExt { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina la extensión para los archivos de log: |
| \* Sugerencia: ($LOGEXT)" |
| read extLog |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $extLog ];then |
| extLog=$LOGEXT |
| fi |
|  |
| validarExtensionLog $extLog |
| done |
|  |
| LOGEXT=$extLog |
| unset cond |
|  |
| } |
|  |
| #Define el directorio del log de auditoria |
| function definirDirLog { |
|  |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de grabación de los logs: |
| \* Sugerencia: ($dirLog)" |
|  |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirLog |
| fi |
| if [ -n $respuesta ];then |
| dirLog=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
|  |
| done |
|  |
| LOGDIR=$GRUPO/$dirLog |
| unset cond |
|  |
| } |
|  |
|  |
| #Define el directorio de reportes |
| function definirDirReportes { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de grabación de los reportes de salida: |
| \* Sugerencia: ($dirReportes)" |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirReportes |
| fi |
| if [ -n $respuesta ];then |
| dirReportes=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
| done |
|  |
| REPODIR=$GRUPO/$dirReportes |
| unset cond |
|  |
|  |
| } |
|  |
|  |
| #Define el directorio de procesados |
| function definirDirProcesados { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de grabación de los archivos externos procesados: |
| \* Sugerencia: ($dirProcesados) " |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirProcesados |
| fi |
| if [ -n $respuesta ];then |
| dirProcesados=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
|  |
| done |
|  |
| PROCDIR=$GRUPO/$dirProcesados |
| unset cond |
|  |
| } |
|  |
| #Define el directorio de aceptados |
| function definirDirAceptados { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de grabación de los archivos externos aceptados: |
| \* Sugerencia: ($dirAceptados) " |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirAceptados |
| fi |
| if [ -n $respuesta ];then |
| dirAceptados=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
|  |
| done |
|  |
| ACEPDIR=$GRUPO/$dirAceptados |
| unset cond |
|  |
| } |
|  |
| #Define el directorio de rechazados |
| function definirDirRechazados { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de grabación de los archivos externos rechazados: |
| \* Sugerencia: ($dirRechazados) " |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirRechazados |
| fi |
| if [ -n $respuesta ];then |
| dirRechazados=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
|  |
| done |
|  |
| RECHDIR=$GRUPO/$dirRechazados |
| unset cond |
|  |
| } |
|  |
| #Valida espacio libre en disco para archivos externos |
| function validarEspacioEnDisco { |
|  |
| espacio\_libre=`df | grep "/\$" | awk '{ print $4 }'` |
|  |
| espacio\_libre\_kb=$[ $espacio\_libre \* 1024 ]   #paso de MB a KB |
| espacio\_requerido=$[ $1 \* 1024 ]   #paso de MB a KB |
|  |
|  |
| if [ $espacio\_requerido -gt $espacio\_libre\_kb ]; then |
|  |
| mensaje=" |
| Insuficiente espacio en disco. |
| Espacio disponible: $espacio\_libre Mb. |
| Espacio requerido $1 Mb |
| Cancele la instalación e inténtelo más tarde o vuelva a intentarlo con otro valor. |
| " |
|  |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
| cond="error" |
| fi |
|  |
| } |
|  |
| #Valida que lo ingresado sea un valor numerico entero y mayor a cero |
| function validarNumerico { |
|  |
| res=`echo "$1" | grep "[^0-9]"` |
| if [ "$1" == "$res" ]; then |
| echo "ERROR -  \"$1\" tiene que ser un número entero." |
| grabarLog "E" "\"$1\" tiene que ser un número entero." |
| cond="error" |
| fi |
| if [ "$1" == "0" ];then |
| echo "ERROR - el valor ingresado no puede ser cero." |
| grabarLog "E" "El valor ingresado no puede ser cero." |
| cond="error" |
| fi |
|  |
| unset res |
| } |
|  |
| #Define el datasize |
| function definirDataSize { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el espacio mínimo libre para el arribo de archivos externos en Mbytes |
| \* Sugerencia: ($DATASIZE)" |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$DATASIZE |
| fi |
|  |
| validarNumerico $respuesta |
|  |
| if [ "$cond"  == "ok" ]; |
| then |
| #11 - Verificar espacio en disco |
| validarEspacioEnDisco $respuesta; |
| fi |
| done |
| unset cond |
|  |
| DATASIZE=$respuesta |
|  |
| } |
|  |
| #Define el directorio de arribos |
| function definirDirArribos { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de arribo de archivos externos: |
| \* Sugerencia: ($dirArribos) " |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirArribos |
| fi |
| if [ -n $respuesta ];then |
| dirArribos=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
| done |
|  |
| ARRIDIR=$GRUPO/$dirArribos |
| unset cond |
|  |
| } |
|  |
|  |
| #Define el directorio de maestros |
| function definirDirMaestros { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de instalación de los archivos maestros: |
| \* Sugerencia: ($dirMaestros)" |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirMaestros |
| fi |
| if [ -n $respuesta ];then |
| dirMaestros=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
| done |
|  |
| MAEDIR=$GRUPO/$dirMaestros |
| unset cond |
|  |
| } |
|  |
| #Valida que no se pueda elegir mas de una vez el mismo directorio |
| function validarListaDirecciones { |
|  |
| for i in ${listaDirecciones[\*]}; do |
| if [ "$i" == "$1" ]; then |
| echo "ERROR - No puede elegir más de una vez el mismo directorio" |
| grabarLog "E" "No puede elegir más de una vez el mismo directorio" |
| cond="error" |
| fi |
| done |
|  |
| listaDirecciones=(${listaDirecciones[\*]} $1) |
|  |
| } |
|  |
|  |
| #Valida que el directorio no exista |
| function validarDirectorios { |
|  |
| if [ -d "$1" ]; then |
| echo "ERROR - el directorio ya existe" |
| grabarLog "E" "El directorio ya existe" |
| cond="error" |
| fi |
|  |
| } |
|  |
| #Define el directorio de ejecutables |
| function definirDirEjecutables { |
|  |
| cond="error" |
|  |
| while [ $cond == "error" ]; do |
| cond="ok" |
| echo " |
| \* Defina el directorio de instalación de los ejecutables: |
| \* Sugerencia: ($dirBinarios)" |
| read respuesta |
|  |
| #Si la persona no ingreso nada entonces se toma por default la sugerencia |
| if [ -z $respuesta ];then |
| respuesta=$dirBinarios |
| fi |
| if [ -n $respuesta ];then |
| dirBinarios=$respuesta |
| fi |
| validarDirectorios $respuesta |
| if [ $cond == "ok" ]; then validarListaDirecciones $respuesta; fi |
|  |
| done |
|  |
| BINDIR=$GRUPO/$dirBinarios |
| unset cond |
|  |
|  |
| } |
|  |
|  |
| function mostrarInformacionInstalacion { |
|  |
| dirconf=`ls $CONFDIR` |
| dirgrupo=`ls $GRUPO` |
|  |
|  |
| mensaje=" |
| Directorio de Trabajo para la instalación: $GRUPO. |
|  |
| Archivos y subdirectorios: |
| $dirgrupo |
|  |
|  |
| Librería del Sistema: $CONFDIR. |
|  |
| Archivos y subdirectorios: |
| $dirconf |
|  |
|  |
| Estado de la instalación: PENDIENTE |
|  |
| Para completar la instalación Ud. deberá: |
|  |
| \*Definir el directorio de instalación de los ejecutables. |
| \*Definir el directorio de instalación de los archivos maestros. |
| \*Definir el directorio de arribo de archivos externos. |
| \*Definir el espacio mínimo libre para el arribo de archivos externos. |
| \*Definir el directorio de grabación de los archivos externos rechazados. |
| \*Definir el directorio de grabación de los archivos externos aceptados. |
| \*Definir el directorio de grabación de los archivos externos procesados. |
| \*Definir el directorio de grabación de los reportes de salida. |
| \*Definir el directorio de grabación de los logs de auditoría. |
| \*Definir la extensión y tamaño máximo para los archivos de log. |
|  |
| " |
|  |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
|  |
| } |
|  |
| #Inicializa las variables utilizadas para definir los directorios |
| function inicializarVariablesADefinir { |
|  |
| dirBinarios="bin" |
| dirMaestros="mae" |
| dirArribos="arribos" |
| dirAceptados="aceptados" |
| dirRechazados="rechazados" |
| dirProcesados="procesados" |
| dirReportes="reportes" |
| dirLog="log" |
|  |
| } |
|  |
| #Modifica el archivo de configuración. |
| function modificarArchivoConfiguracion { |
|  |
| echo " |
| Actualizando la configuración del sistema..." |
|  |
| #grabarLog "INFORMATIVO" "Actualizando la configuración del sistema..." |
|  |
| fecha=`date +%d/%m/%y` |
| hora=`date +%R` |
|  |
| fecha="${fecha} $hora" |
|  |
| usuario=`who | awk '{ print $1 }'` |
|  |
| GRUPO="${GRUPO}=$usuario=$fecha" |
| BINDIR="${BINDIR}=$usuario=$fecha" |
| MAEDIR="${MAEDIR}=$usuario=$fecha" |
| ARRIDIR="${ARRIDIR}=$usuario=$fecha" |
| ACEPDIR="${ACEPDIR}=$usuario=$fecha" |
| RECHDIR="${RECHDIR}=$usuario=$fecha" |
| PROCDIR="${PROCDIR}=$usuario=$fecha" |
| REPODIR="${REPODIR}=$usuario=$fecha" |
| LOGDIR="${LOGDIR}=$usuario=$fecha" |
| LOGEXT="${LOGEXT}=$usuario=$fecha" |
| LOGSIZE="${LOGSIZE}=$usuario=$fecha" |
| DATASIZE="${DATASIZE}=$usuario=$fecha" |
|  |
|  |
| echo " |
| GRUPO=$GRUPO |
| BINDIR=$BINDIR |
| MAEDIR=$MAEDIR |
| ARRIDIR=$ARRIDIR |
| ACEPDIR=$ACEPDIR |
| RECHDIR=$RECHDIR |
| PROCDIR=$PROCDIR |
| REPODIR=$REPODIR |
| LOGDIR=$LOGDIR |
| LOGEXT=$LOGEXT |
| LOGSIZE=$LOGSIZE |
| DATASIZE=$DATASIZE |
| "> $CONFDIR/$confFile |
|  |
| } |
|  |
| #Da permisos de ejecucion a los archivos |
| function darPermisosDeEjecucion { |
|  |
| chmod 777 $BINDIR/\* |
|  |
| } |
|  |
| function darPermisosDeEjecucionInicial { |
|  |
| chmod 777 $grupo/instalacion/bin/\* |
|  |
| } |
|  |
| #Mueve archivos maestros al directorio MAEDIR. |
| #Mueve la tabla de separadores y la tabla de campos al directorio CONFDIR |
| #Mueve los ejecutables y funciones al directorio BINDIR |
| function moverArchivos { |
|  |
| echo " |
| Instalando Archivos Maestros..." |
|  |
| grabarLog "INFORMATIVO" "Instalando Archivos Maestros..." |
|  |
| for i in PPI.mae p-s.mae |
| do |
| if [ -f "$grupo/instalacion/mae/$i" ]; then |
|  |
| MoverX  "$grupo/instalacion/mae/$i" "$MAEDIR" "InstalarX.sh" |
|  |
| resultado=$? |
|  |
| #echo "RESULTADO: $resultado" |
|  |
| if [ "$resultado" -ne 0 ]; then |
|  |
| mensaje="Ha ocurrido un error al mover $i" |
|  |
| echo "$mensaje" |
|  |
| grabarLog "E" "$mensaje" |
| fi |
|  |
| else |
| echo -e "El comando $i no existe\n" |
| fi |
| done |
|  |
| echo " |
| Instalando Tablas de Configuración..." |
|  |
| grabarLog "INFORMATIVO" "Instalando Tablas de Configuración..." |
|  |
| for i in T1.tab T2.tab |
| do |
| if [ -f "$grupo/instalacion/tablas/$i" ]; then |
|  |
| MoverX  "$grupo/instalacion/tablas/$i" "$CONFDIR" "InstalarX.sh" |
|  |
| resultado=$? |
|  |
| #echo "RESULTADO: $resultado" |
|  |
| if [ "$resultado" -ne 0 ]; then |
|  |
| mensaje="Ha ocurrido un error al mover $i" |
|  |
| echo "$mensaje" |
|  |
| grabarLog "E" "$mensaje" |
| fi |
|  |
| else |
| echo -e "El comando $i no existe\n" |
| fi |
| done |
|  |
| echo " |
| Instalando Programas y Funciones..." |
|  |
| grabarLog "I" "Instalando Programas y Funciones..." |
|  |
| #Mover.sh debe ser el ultimo en la lista de ejecutables (se mueve a si mismo) |
| EJECUTABLES=(InicioX.sh DetectaX.sh Interprete.sh ReporteX.pl GlogX.sh VlogX.sh StartX.sh StopX.sh valPais.sh MoverX.sh) |
|  |
| for i in ${EJECUTABLES[\*]} |
| do |
| if [ -f "$grupo/instalacion/bin/$i" ]; then |
|  |
| MoverX  "$grupo/instalacion/bin/$i" "$BINDIR" "InstalarX.sh" |
|  |
| resultado=$? |
|  |
| #echo "RESULTADO: $resultado" |
|  |
| if [ "$resultado" -ne 0 ]; then |
|  |
| mensaje="Ha ocurrido un error al mover $i" |
|  |
| echo "$mensaje" |
|  |
| grabarLog "E" "$mensaje" |
| fi |
|  |
| else |
| echo -e "El comando $i no existe\n" |
| fi |
| done |
| } |
|  |
| #Crea los directorios que no existen |
| function crearEstructurasDeDirectorios { |
|  |
| echo "Creando Estructuras de Directorio.... |
| " |
| grabarLog "I" "Creando Estructuras de Directorio...." |
|  |
| listaDirectorios=( $BINDIR $MAEDIR $ARRIDIR $ACEPDIR $RECHDIR $PROCDIR $REPODIR $LOGDIR) |
|  |
| for i in ${listaDirectorios[\*]}; do |
| echo "Creando $i..." |
| grabarLog "I" "Creando $i..." |
| mkdir -p $i |
| done |
|  |
| } |
|  |
| #Muestra los path seteados para el archivo de configuracion |
| function mostrarPaths { |
|  |
| mensaje=" |
| TP SO7508 1er cuatrimestre 2013. Tema T Copyright (c) Grupo 01 |
|  |
| \* Directorio de Trabajo: $GRUPO |
| \* Librería del Sistema:  $CONFDIR |
| \* Directorio de grabación de los ejecutables: $BINDIR |
| \* Directorio de instalación de los archivos maestros: $MAEDIR |
| \* Directorio de arribo de archivos externos: $ARRIDIR |
| \* Espacio mínimo libre para el arribo de archivos externos: $DATASIZE Mb |
| \* Directorio de grabación de los archivos externos aceptados: $ACEPDIR |
| \* Directorio de grabación de los archivos externos rechazados: $RECHDIR |
| \* Directorio de grabación de los archivos externos procesados: $PROCDIR |
| \* Directorio de grabación de los reportes de salida: $REPODIR |
| \* Directorio de grabación de los logs de auditoría: $LOGDIR |
| \* Extensión para los archivos de Log: $LOGEXT |
| \* Tamaño máximo para los archivos de log: $LOGSIZE Kb |
|  |
| " |
|  |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
|  |
| } |
|  |
|  |
| #Valida si el usuario tiene instalado perl. De ser asi muestra la version del mismo. En caso contrario muestra |
| #un mensaje y sale de la instalacion. |
| function validarPerl { |
|  |
| pathPerl=`which perl` |
| validacionPerl="false" |
|  |
| if [ "$?" == 0 ]; then |
| #6.2 - Perl esta instalado. Mostrar y grabar en el log un mensaje informativo con la version de Perl que se encuentra instalada. |
| version=` $pathPerl -v | grep "v[5-9]\."` |
| if [ -n "$version" ]; then |
| validacionPerl="true" |
| mensaje=" |
| TP SO7508 1er cuatrimestre 2013. Tema T Copyright (c) Grupo 01. |
| Perl Version: $version |
| " |
| echo "$mensaje" |
|  |
| grabarLog "I" "$mensaje" |
|  |
| fi |
| fi |
|  |
| if [ $validacionPerl == "false" ]; then |
| #6.1 - Validacion de Perl da error. Mostrar y grabar en el log. |
| mensaje=" |
| TP S07508 1er cuatrimestre 2013. Tema T Copyright (C) Grupo 01. |
| Para instalar el TP es necesario contar con  Perl 5 o superior instalado. Efectúe su instalación e inténtelo nuevamente. |
|  |
| Proceso de Instalación Cancelado. |
| " |
| echo "$mensaje" |
|  |
| grabarLog "I" "$mensaje" |
|  |
| exit 0 |
| fi |
|  |
| } |
|  |
| #Valida que la respuesta sea S o N |
| function validarRespuesta { |
|  |
| respuesta="" |
|  |
| while [ "$respuesta" != "s" ] |
| do |
| if [ "$respuesta" == "n" ]; then |
| echo "\* El proceso de instalación ha sido CANCELADO. \*" |
| grabarLog "I" "\* El proceso de instalación ha sido CANCELADO. \*" |
| exit 0 |
| fi |
| echo "$1" |
| read respuesta |
| done |
|  |
| } |
|  |
| function mostrarComponentesInstalados { |
|  |
| dirmae="" |
| dirbin="" |
| faltanej="" |
| faltanmae="" |
| faltaarr="" |
| faltaacep="" |
| faltarech="" |
| faltaproc="" |
| faltarepo="" |
| faltalog="" |
| faltaext="" |
| faltatamaniolog="" |
| faltatamanio="" |
|  |
|  |
| if [ "$ejecutables" -ne "0" ]; |
| then |
| dirbin=`ls $BINDIR` |
| fi |
|  |
| if [ "$maestros" -ne "0" ]; |
| then |
| dirmae=`ls $MAEDIR` |
| fi |
|  |
|  |
| if [ "$ejecutables" -eq "0" ]; |
| then |
| faltanej="Faltan todos los archivos ejecutables necesarios, se copiaran los mismos del directorio de instalación." |
| fi |
|  |
|  |
| if [ "$maestros" -eq "0" ]; |
| then |
| faltanmae="Faltan todos los archivos maestros necesarios, se copiaran los mismos del directorio de instalación." |
|  |
| fi |
|  |
| if [ $arribos -gt 0 ]; |
| then |
|  |
| faltaarr="No se encontró el directorio de arribos especificado en el archivo de configuracion. |
| Se toma por default:  ARRIDIR=\"$ARRIDIR\"" |
|  |
| fi |
|  |
| if [ $aceptados -gt 0 ]; |
| then |
|  |
| faltaacep="No se encontró el directorio de los archivos aceptados especificado en el archivo de configuracion. |
| Se toma por default:  ACEPDIR=\"$ACEPDIR\"" |
|  |
| fi |
|  |
| if [ $rechazados -gt 0 ]; |
| then |
|  |
| faltarech="No se encontró el directorio de los archivos rechazados especificado en el archivo de configuracion. |
| Se toma por default:  RECHDIR=\"$RECHDIR\"" |
|  |
| fi |
|  |
| if [ $procesados -gt 0 ]; |
| then |
|  |
| faltaproc="No se encontró el directorio de los archivos procesados especificado en el archivo de configuracion. |
| Se toma por default:  PROCDIR=\"$PROCDIR\"" |
|  |
| fi |
|  |
| if [ $reportes -gt 0 ]; |
| then |
|  |
| faltarepo="No se encontró el directorio de los archivos de reporte especificado en el archivo de configuracion. |
| Se toma por default:  REPODIR=\"$REPODIR\"" |
|  |
| fi |
|  |
| if [ $log -gt 0 ]; |
| then |
|  |
| faltalog="No se encontró el directorio del archivo de Log especificado en el archivo de configuracion. |
| Se toma por default:  LOGDIR=\"$LOGDIR\"" |
|  |
| fi |
|  |
| if [ $extension -gt 0 ]; |
| then |
|  |
| faltaext="No se encontró la extensión del archivo de log. |
| Se toma por default:  LOGEXT=$LOGEXT" |
|  |
| fi |
|  |
| if [ $tamanioLog -gt 0 ]; |
| then |
|  |
| faltatamaniolog="No se encontró el tamaño máximo del archivo de Log. |
| Se toma por default:  LOGSIZE=$LOGSIZE" |
|  |
| fi |
|  |
| if [ $tamanio -gt 0 ]; |
| then |
|  |
| faltatamanio="No se encontró el tamaño mínimo de espacio en disco para la aplicación. |
| Se toma por default:  DATASIZE=$DATASIZE" |
|  |
| fi |
|  |
|  |
| mensaje=" |
| TP SO7508 1er cuatrimestre 2013. Tema T Copyright (c) Grupo 01. |
|  |
| Componentes existentes: |
|  |
| Directorio de instalación de los ejecutables: $BINDIR |
|  |
| Archivos: |
|  |
| $dirbin |
|  |
| Directorio de instalación de los archivos maestros: $MAEDIR |
|  |
| Archivos: |
|  |
| $dirmae |
|  |
| Componentes faltantes: |
|  |
| $faltanej |
| $faltanmae |
| $faltaarr |
| $faltaacep |
| $faltarech |
| $faltaproc |
| $faltarepo |
| $faltalog |
| $faltaext |
| $faltatamaniolog |
| $faltatamanio |
|  |
| " |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
|  |
| } |
|  |
|  |
| function mostrarMensajeInstalacionFinalizada { |
|  |
| dirconf=`ls $CONFDIR` |
| dirbin=`ls $BINDIR` |
| dirmae=`ls $MAEDIR` |
|  |
| mensaje=" |
| TP SO7508 Primer Cuatrimestre 2013. Tema X Copyright (c) Grupo 01. |
|  |
| Librería del sistema: $CONFDIR |
|  |
| Archivos: |
| $dirconf |
|  |
|  |
| Ejecutables: $BINDIR |
|  |
| Archivos: |
| $dirbin |
|  |
|  |
| Archivos maestros: $MAEDIR |
|  |
| Archivos: |
| $dirmae |
|  |
|  |
| Directorio de arribo de archivos externos: $ARRIDIR |
|  |
| Archivos externos aceptados: $ACEPDIR |
|  |
| Archivos externos rechazados: $RECHDIR |
|  |
| Archivos procesados: $PROCDIR |
|  |
| Reportes de salida: $REPODIR |
|  |
| Logs de auditoría del Sistema: $LOGDIR/InstalarX.$LOGEXT |
|  |
| Estado de la instalación: COMPLETA |
|  |
| Proceso de Instalación CANCELADO. |
| " |
|  |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
|  |
| } |
|  |
| #Completa la instalacion en caso de haber encontrado una ya existente (incompleta) |
| function completarInstalacion { |
|  |
| mostrarComponentesInstalados |
|  |
| echo " |
| Estado de la instalación: INCOMPLETA " |
|  |
| grabarLog "I" "Estado de la instalación: INCOMPLETA " |
|  |
| validarRespuesta "Desea completar la instalación? [s/n]" |
|  |
| #4.3 - El usuario indico SI por lo que paso a chequear que Perl este instalado |
| validarPerl |
|  |
| #4.3.3 - Mostrar los valores de instalacion |
| mostrarPaths |
|  |
| echo "Estado de la instalación: LISTA" |
|  |
| grabarLog "I" "Estado de la instalación: LISTA" |
|  |
| #20 - Confirmar Inicio de Instalacion |
| validarRespuesta "Iniciando Instalación. ¿Está UD. seguro? [s/n]" |
|  |
| #21 - Instalacion |
| crearEstructurasDeDirectorios |
|  |
| #21.2 - Mueve archivos maestros al directorio MAEDIR. |
| #21.3 - Mueve la tabla de separadores y la tabla de campos al directorio CONFDIR |
| #21.4 - Mueve los ejecutables y funciones al directorio BINDIR |
| moverArchivos |
|  |
| darPermisosDeEjecucion |
|  |
| # 21.5 - Actualizar el archivo de configuracion. |
| modificarArchivoConfiguracion |
|  |
| #22 - Borrar los archivos temporarios, si los hubiese generado. NO APLICA |
| } |
|  |
| #Verifica la existencia de una cantidad de Xmb disponibles para la instalacion del tp. Si no lo esta asigna 100mb. |
| function chequearDATASIZE { |
|  |
| if [ -z "$DATASIZE" ]; then |
| DATASIZE=100 |
| tamanio=1 |
| cantErrores=$[ $cantErrores + 1 ] |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de una cantidad de X kb de longitud para el archivo de log. Si no lo esta asigna 400kb. |
| function chequearLOGSIZE { |
|  |
| if [ -z "$LOGSIZE" ]; then |
| LOGSIZE=400 |
| tamanioLog=1 |
| cantErrores=$[ $cantErrores + 1 ] |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de reportes  definido en el archivo de configuracion |
| function chequearREPODIR { |
|  |
| if [ -z "$REPODIR" ]; then |
| REPODIR="$grupo/reportes" |
| reportes=1 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$REPODIR" ]; then |
| reportes=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la extension del archivo de log |
| function chequearLOGEXT { |
|  |
| if [ -z "$LOGEXT" ]; then |
| LOGEXT=".log" |
| extension=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de archivos de log  definido en el archivo de configuracion |
| function chequearLOGDIR { |
|  |
| if [ -z "$LOGDIR" ]; then |
| LOGDIR="$grupo/log" |
| log=1 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$LOGDIR" ]; then |
| log=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de archivos maestros definido en el archivo de configuracion |
| function chequearMAEDIR { |
|  |
| if [ -z "$MAEDIR" ]; then |
| MAEDIR="$grupo/mae" |
| maestros=0 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$MAEDIR" ]; then |
| maestros=0 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de procesados definido en el archivo de configuracion |
| function chequearPROCDIR { |
|  |
| if [ -z "$PROCDIR" ]; then |
| PROCDIR="$grupo/procesados" |
| procesados=1 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$PROCDIR" ]; then |
| procesados=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de rechazados definido en el archivo de configuracion |
| function chequearRECHDIR { |
|  |
| if [ -z "$RECHDIR" ]; then |
| RECHDIR="$grupo/rechazados" |
| rechazados=1 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$RECHDIR" ]; then |
| rechazados=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de aceptados definido en el archivo de configuracion |
| function chequearACEPDIR { |
|  |
| if [ -z "$ACEPDIR" ]; then |
| ACEPDIR="$grupo/aceptados" |
| aceptados=1 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$ACEPDIR" ]; then |
| aceptados=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de arribos definido en el archivo de configuracion |
| function chequearARRIDIR { |
|  |
| if [ -z "$ARRIDIR" ]; then |
| ARRIDIR="$grupo/arribos" |
| arribos=1 |
| cantErrores=$[ $cantErrores + 1] |
| else |
| if ! [ -d "$ARRIDIR" ]; then |
| arribos=1 |
| cantErrores=$[ $cantErrores + 1] |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica la existencia de directorio de archivos ejecutables definido en el archivo de configuracion |
| function chequearBINDIR { |
|  |
| if [ -z "$BINDIR" ]; then |
| BINDIR="$grupo/bin" |
| cantErrores=$[ $cantErrores + 1] |
| ejecutables=0 |
| else |
| if ! [ -d "$BINDIR" ]; then |
| cantErrores=$[ $cantErrores + 1] |
| ejecutables=0 |
| fi |
| fi |
|  |
| } |
|  |
| #Verifica cuales son los componentes ya instalados |
| function chequearComponentesInstalados { |
|  |
| chequearBINDIR |
| chequearMAEDIR |
| chequearARRIDIR |
| chequearACEPDIR |
| chequearRECHDIR |
| chequearPROCDIR |
| chequearREPODIR |
| chequearLOGDIR |
| chequearLOGEXT |
| chequearLOGSIZE |
| chequearDATASIZE |
|  |
| } |
|  |
| #Muestra mensaje inicio instalacion |
| function mostrarMensajeInicioInstalacion { |
| echo ' |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| \* TP SO7508 Primer Cuatrimestre 2013. Tema X Copyright (c) Grupo 01 |
| \* A T E N C I O N: Al instalar TP SO7508 Primer Cuatrimestre 2013 UD. |
| \* expresa aceptar los términos y condiciones del "ACUERDO DE LICENCIA DE |
| \* SOFTWARE" incluído en este paquete. |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| ' |
| } |
|  |
| #Comienza la instalcion |
| function instalar { |
|  |
| #5.0 - Mostrar mensaje de acuerdo de licencia de software |
| mostrarMensajeInicioInstalacion |
|  |
| validarRespuesta "¿Desea continuar con la instalación? [s/n]" |
|  |
| #Son las sugerencias que se dan al usuario a la hora de definir los directorios |
| inicializarVariablesADefinir |
|  |
| while [ "$respuesta1" != "s" ] |
| do |
| clear |
|  |
| #6 - Chequear que Perl este instalado |
| validarPerl |
|  |
| mostrarInformacionInstalacion |
|  |
| listaDirecciones=() |
|  |
| #7 - Definir el directorio de instalacion de los ejecutables |
| definirDirEjecutables |
|  |
| #8 - Definir el directorio de instalacion de los archivos maestros |
| definirDirMaestros |
|  |
| #9 - Definir el directorio de arribo de archivos externos |
| definirDirArribos |
|  |
| #10 - Definir el espacio minimo libre para el arribo de archivos externos |
| # Dentro de definirDataSize verifico espacio en disco (11 - Verificar espacio en disco) |
| definirDataSize |
|  |
| #12 - Definir el directorio de grabacion de los archivos rechazados |
| definirDirRechazados |
|  |
| #13 - Definir el directorio de grabacion de los archivos aceptados |
| definirDirAceptados |
|  |
| #14 - Definir el directorio de trabajo principal del proceso interprete |
| definirDirProcesados |
|  |
| #15 - Definir el directorio de grabacion de los reportes de salida |
| definirDirReportes |
|  |
| #16 - Definir el directorio de logs para los comandos |
| definirDirLog |
|  |
| #17 - Definir la extension para los archivos de log |
| definirLogExt |
|  |
| #18 - Definir el tamanio maximo para los archivos de log |
| definirLogSize |
|  |
| clear |
|  |
| #19 - Mostrar estructura de directorios resultante y valores de parametros de configuracion |
| mostrarPaths |
|  |
| echo "Estado de la instalación: LISTA" |
|  |
| grabarLog "I" "Estado de la instalación: LISTA" |
|  |
| echo "¿Los datos ingresados son correctos? [s/n]" |
|  |
| read respuesta1 |
| done |
|  |
| #20 - Confirmar inicio de instalacion |
| validarRespuesta "Iniciando Instalación. Está UD. seguro? [s/n]" |
|  |
| #21 - Instalacion |
| #21.1 - Crear las estructuras de directorio requeridas. |
| crearEstructurasDeDirectorios |
|  |
| #21.2 - Mover los archivos maestros al directorio MAEDIR |
| #21.3 - Mover la tabla de separadores y la tabla de campos al directorio CONFDIR |
| #21.4 - Mover los ejecutables y funciones al directorio BINDIR |
| moverArchivos |
|  |
| darPermisosDeEjecucion |
|  |
| #21.5 - Actualizar el archivo de configuracion |
| modificarArchivoConfiguracion |
|  |
| #22 - Borrar archivos temporarios, si los hubiese generado. NO APLICA |
| } |
|  |
| #Inicializa las variables del directorio del archivo de configuracion |
| function inicializarVariablesDefault { |
|  |
| GRUPO=$grupo |
| CONFDIR=$GRUPO/conf |
| BINDIR=$GRUPO/bin |
| MAEDIR=$GRUPO/mae |
| ARRIDIR=$GRUPO/arribos |
| ACEPDIR=$GRUPO/aceptados |
| RECHDIR=$GRUPO/rechazados |
| PROCDIR=$GRUPO/procesados |
| REPODIR=$GRUPO/reportes |
| LOGDIR=$GRUPO/log |
| LOGEXT=.log |
| # Maximo tamanio de archivo de log = 400kb |
| LOGSIZE=400 |
| # 100mb de espacio libre para archivos externos |
| DATASIZE=100 |
|  |
| } |
|  |
| #Llama al log para grabar |
| #$2 = mensaje $1 = tipo (error, informativo, warning, etc) |
| function grabarLog { |
|  |
| export CONFDIR |
| GlogX "InstalarX.sh" "$1" "$2" "InstalarX" |
|  |
| } |
|  |
| #Crea el directorio y el archivo de configuracion si no existe |
| function crearDirectorioArchivoConfiguracion { |
|  |
| mkdir -p $CONFDIR |
| >>$CONFDIR/$confFile |
|  |
| } |
|  |
| #Crea el archivo de log |
| function crearLog { |
|  |
| >>$CONFDIR/$logFile |
| chmod 777 $CONFDIR/$logFile |
|  |
| } |
|  |
| #Graba mensajes de inicio en el log |
| function mensajesInicioLog { |
|  |
| mensaje="Inicio de Ejecución" |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
| mensaje="Log del Comando InstalarX: $CONFDIR/$logFile" |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
| mensaje="Directorio de Configuración: $CONFDIR" |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
|  |
| } |
|  |
| #Chequea la existencia de archivos maestros en el directorio maestros para poder efectuar la instalación. |
| function testArchivosMaestros  { |
|  |
| MAESTROS=(PPI.mae p-s.mae) |
|  |
| for i in ${MAESTROS[\*]}; do |
| if ! [ -f $1/"$i" ]; then |
| echo "\* Falta el archivo maestro: \"$i\"" |
| cantErrores=$[ $cantErrores + 1 ] |
| grabarLog "SE" "\* Falta el archivo maestro: \"$i\"" |
| fi |
| done |
|  |
| } |
|  |
|  |
| #Chequea la existencia de los archivos ejecutables en el directorio de ejecutables para poder efectuar la instalación. |
| function testComandos  { |
|  |
| EJECUTABLES=(InicioX.sh DetectaX.sh Interprete.sh ReporteX.pl GlogX.sh VlogX.sh StartX.sh StopX.sh MoverX.sh) |
|  |
| for i in ${EJECUTABLES[\*]}; do |
| if ! [ -f $1/"$i" ]; then |
| echo "\* Falta el ejecutable: \"$i\"" |
| cantErrores=$[ $cantErrores + 1 ] |
| grabarLog "SE" "\* Falta el archivo maestro: \"$i\"" |
| fi |
| done |
|  |
| } |
|  |
| #Verifico la existencia de los archivos necesarios para realizar la instalacion. Maestros y Ejecutables. |
| function verificarArchivosInstalacion { |
|  |
|  |
| cantErrores=0 |
|  |
| testComandos "$grupo/instalacion/bin" |
| testArchivosMaestros "$grupo/instalacion/mae" |
|  |
| if [ $cantErrores -gt 0 ]; then |
| mensaje=" |
| Proceso de Instalación cancelado. |
| Asegúrese de que contar con los archivos arriba mencionados y vuelva a intentarlo." |
| echo $mensaje |
| grabarLog "SE" $mensaje |
| exit 0 |
| fi |
|  |
| } |
|  |
|  |
| function main { |
|  |
| clear |
| cd .. |
|  |
| #grupo: path donde se encuentra el tp ../grupo01/ |
| grupo="$PWD" |
|  |
| source "$grupo/instalacion/bin/GlogX.sh"; |
| source "$grupo/instalacion/bin/MoverX.sh"; |
|  |
| #CONFDIR: ubicacion del directorio de configuracion |
| CONFDIR="$grupo/conf" |
|  |
| #confFile: nombre archivo configuracion |
| confFile="InstalarX.conf" |
|  |
| #logFile: nombre archivo de log |
| logFile="InstalarX.log" |
|  |
| echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
| \*   Bienvenido al Asistente de instalación del sistema ControlX        \* |
| \*   TP SO7508 Primer Cuatrimestre 2013. Tema X Copyright (c) Grupo 01  \* |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" |
|  |
| crearDirectorioArchivoConfiguracion |
| #1 - Inicializar archivo de log |
| crearLog |
|  |
| darPermisosDeEjecucionInicial |
|  |
| #Verifico la existencia de archivos maestros y de los ejecutables |
| verificarArchivosInstalacion |
|  |
| #2 - Mostrar (y grabar en el log) donde se graba el log de la instalacion |
| #3 - Mostrar (y grabar en el log) el nombre del directorio de configuracion |
| mensajesInicioLog |
|  |
| #4 - Detectar si el paquete ControlX o alguno de sus componentes ya esta instalado |
| # Cuento la cantidad de lines del archivo de configuracion |
| cantLineas=$(wc -l $CONFDIR/$confFile | awk '{print $1}') |
|  |
| #Si tiene 0 lineas el archivo de configuración es porque no tiene nada instalado |
| if [ "$cantLineas" -eq "0" ]; |
| then |
| #5 - Aceptacion de terminos y condiciones |
| inicializarVariablesDefault |
| instalar |
| else |
| #4.0 - Verifico instalacion previa |
| leerVariablesDeConfiguracion "$CONFDIR/$confFile" |
|  |
| cantErrores=0 |
| maestros=1 |
| ejecutables=1 |
| arribos=0 |
| aceptados=0 |
| rechazados=0 |
| procesados=0 |
| reportes=0 |
| log=0 |
| extension=0 |
| tamanio=0 |
| tamanioLog=0 |
|  |
| chequearComponentesInstalados |
|  |
| if [ $cantErrores -gt 0 ]; then |
| #4.2 - Si falta algun componente |
| completarInstalacion |
| Else |
| #4.1 - Si esta completo |
| mostrarMensajeInstalacionFinalizada |
| Fi |
| Fi |
|  |
| #23 - Mostrar mensaje de fin de instalacion |
| echo " |
| Instalación Finalizada" |
| #grabarLog "INFORMATIVO" "Instalacion Finalizada" |
|  |
| #24 – FIN |
|  |
|  |
| } |
|  |
| main |

**4.2 - InicioX.sh**

**Título:** InicioX.sh

**Justificación**

Este comando sirve para que los demas scripts puedan ejecturse correctamente, chequeando de que existan y tengan los permisos adecuados, asi tambien tiene en cuenta que existan los archivos maestros y demas necesarios para la ejecucion del ControlX. Setea las variables de ambiente para evitar el harcode de directorios, y tambien setea la variable PATH para que los demas scripts se puedan ejecutar desde cualquier otro script o por consola.

**Archivos de Input, Intermedios y de Output**

Archivos Input:

InstalX.conf (de aquí se obtiene las rutas a los directorios ya se ARRIDIR, PROCDIR, etc)

Archivos Output:

InicioX.sh crea un log en $LOGDIR: InicioX.log

**Parámetros y Opciones**

No posee parámetros.

**Invocación Manual y/o automática con ejemplos**

Para la invocacion se debe ejecutar el comando manualmente y de la forma:

“ . InicioX.sh” (ejecución con punto espacio)

**Hipótesis y Aclaraciones especificas del comando**

* Para ejecutar el demonio "DetectaX.sh" se usa el comando "StartX.sh".
* Primeros se setean todas las variables de entorno para luego poder chequear que existan archivos y tengan los permisos requeridos.
* El directorio CONFDIR es conocido.
* Para ejecutar el comando GlogX.sh a la hora de grabar en el log los mensajes de informacion, se tiene en cuenta que el comando no tiene los permisos de ejecucion para poder realizar la tarea, con lo cual se hacen primero los chequeos/seteos correspondientes y despues se ejecuta el mismo, al igual que el DetectaX.sh. No como lo indica el enunciado en los pasos sugeridos.

**Código fuente**

|  |
| --- |
| #!/bin/bash |
|  |
| #=========================================================== |
| # ARCHIVO: InicioX.sh |
| # |
| # FIRMA: . InicioX.sh |
| # |
| # DESCRIPCION: Comando encargado de inicializar el ambiente para correr CONTROLX |
| # \ |
| # AUTOR: De Zan, Ignacio. |
| # PADRON: 91525 |
| # |
| #=========================================================== |
|  |
| # Llama al log para grabar |
| # $2 = mensaje $1 = tipo (error, informativo, warning, etc) |
|  |
| function grabarLog { |
|  |
| source GlogX.sh; GlogX "InicioX.sh" "$1" "$2" "InicioX" |
|  |
| } |
|  |
| # Chequea que existan los comandos en la carpeta BINDIR, |
| # y tengan los permisos de ejecucion seteados,sino los setea. |
|  |
| function chequearComandos { |
|  |
| for i in ${comandos[\*]} |
| do |
| if [ -f $BINDIR/$i ]; then |
| echo -e "El comando $i existe\n" |
|  |
| if [ -x $BINDIR/$i ]; then |
| echo -e "y tiene permisos de ejecucion\n" |
| else |
| chmod 777 $BINDIR/$i |
| echo -e "`ls -l $BINDIR/$i` \n" |
| fi |
|  |
| else |
| echo -e "El comando $i no existe\n" |
| fi |
| done |
| } |
|  |
| # Chequea que existan los maestros en la carpeta MAEDIR, |
| # y tengan los permisos de lectura seteados,sino los setea. |
|  |
| function chequearMaestros { |
|  |
| for i in PPI.mae p-s.mae |
| do |
| if [ -f $MAEDIR/$i ]; then |
| echo -e "El archivo maestro $i existe\n" |
|  |
| if [ -r $MAEDIR/$i ] &&  ! [ -w $MAEDIR/$i ]; then |
| echo -e "y tiene permisos de lectura, pero no escritura\n" |
| else |
| chmod 444 $MAEDIR/$i |
| echo -e `ls -l $MAEDIR/$i` |
| fi |
|  |
| else |
| echo -e "El archivo maestro $i no existe\n" |
| fi |
| done |
|  |
| } |
|  |
| # Chequea que existan las tablas en la carpeta CONFDIR, |
| # y tengan los permisos de lectura seteados,sino los setea. |
|  |
| function chequearTablas { |
|  |
| for i in T2.tab T1.tab |
| do |
| if [ -f $CONFDIR/$i ]; then |
| echo -e "El archivo maestro $i existe\n" |
|  |
| if [ -r $CONFDIR/$i ] &&  ! [ -w $CONFDIR/$i ]; then |
| echo -e "y tiene permisos de lectura, pero no escritura\n" |
| else |
| chmod 444 $CONFDIR/$i |
| echo -e `ls -l $CONFDIR/$i` |
| fi |
|  |
| else |
| echo -e "El archivo maestro $i no existe\n" |
| fi |
| done |
|  |
| } |
|  |
| # Lee las variables de Config del archivo InstalX.conf |
|  |
| function setVariablesDeConfiguracion { |
|  |
| export $2=`grep "$2" "$1" | cut -d"=" -f 2` |
| } |
|  |
| #Verifica que las variables de ambiente este seteadas |
|  |
| function chequearVarAmbiente { |
|  |
| for var in ${variables[\*]} |
| do |
| res=`env | grep $var | cut -d"=" -f 2` |
|  |
| if [ -z "$res" ]; then |
|  |
| echo -e "Falta la variable de ambiente $var, agregando...\n" |
|  |
| setVariablesDeConfiguracion $CONFDIR/$confFile $var |
|  |
| echo -e "Variable $var ahora esta agregada\n" |
|  |
| else |
| echo -e "La variable de ambiente $var=$res existe\n" |
| fi |
| done |
| } |
|  |
| # Chequea que la carpeta donde se encuentran los comandos, este incluido en la variable PATH, |
| # para su correcta ejecucion, sino lo setea |
|  |
| function chequearPaths { |
|  |
| ejec=`echo $PATH | grep $BINDIR` |
|  |
| if [ -z "$ejec" ]; then |
|  |
| echo -e "No esta el path de ejecutables, agregando...\n" |
|  |
| export PATH=$PATH:$BINDIR |
|  |
| echo -e "Agregado\n" |
|  |
| else |
|  |
| echo -e "El path de ejecutables esta seteado\n" |
|  |
| fi |
| } |
|  |
| #Valida que lo ingresado sea un valor numerico entero y mayor a cero |
| function validarNumerico { |
|  |
| res=`echo "$1" | grep "[^0-9]"` |
| if [ "$1" == "$res" ]; then |
| echo "ERROR -  \"$1\" tiene que ser un número entero." |
| #grabarLog "E" "\"$1\" tiene que ser un número entero." |
| cond="error" |
| fi |
| if [ "$1" == "0" ];then |
| echo "ERROR - el valor ingresado no puede ser cero." |
| #grabarLog "E" "El valor ingresado no puede ser cero." |
| cond="error" |
| fi |
|  |
| unset res |
| } |
|  |
|  |
| # Funcion que pide por teclado la cantidad de loops que quiere que haga el DetectaX |
|  |
| function ingresarCantLoop { |
|  |
| cond="error" |
| while [ $cond == "error" ] |
| do |
| echo "Cantidad de ciclos de DetectaX ? (100 ciclos)" |
| read CANLOOP |
| cond="ok" |
| validarNumerico $CANLOOP |
|  |
| done |
| unset cond |
|  |
| } |
|  |
| # Funcion que pide por teclado el tiempo de espera que quiere que tenga el DetectaX |
|  |
| function ingresartEspera { |
|  |
| cond="error" |
| while [ $cond == "error" ] |
| do |
| echo "Tiempo de espera entre ciclos? (1 minuto)" |
| read TESPERA |
| cond="ok" |
| validarNumerico $TESPERA |
|  |
| done |
| unset cond |
|  |
| } |
|  |
| # Chequea si el proceso DetectaX ya esta corriendo |
|  |
| function chequearDetectaX { |
|  |
| resultado=`ps -A | grep "DetectaX.sh"` |
|  |
| if [ -z "$resultado" ]; then |
| return 0 |
| else |
| return 1 |
| fi |
| } |
|  |
| # Pregunta si se desea iniciar el comando DetectaX, y actua segun la respuesta. |
|  |
| function lanzarDetectaX { |
|  |
| echo "Desea efectuar la activación de DetectaX? [s/n]" |
| read resp |
|  |
| while [ "$resp" != "s" ] |
| do |
| if [ "$resp" == "n" ]; then |
| return 1 |
| fi |
|  |
| echo "Ingrese una respuesta valida" |
| read resp |
|  |
| done |
| return 0 |
| } |
|  |
| function mostrarMensajeInstalacionFinalizada { |
|  |
| dirconf=`ls $CONFDIR` |
| dirbin=`ls $BINDIR` |
| dirmae=`ls $MAEDIR` |
|  |
| #procssid=`ps | grep 'DetectaX' | cut -d" " -f2` |
|  |
| mensaje=" |
| TP SO7508 Primer Cuatrimestre 2013. Tema X Copyright (c) Grupo 01. |
|  |
| Librería del sistema: $CONFDIR |
|  |
| Archivos: |
| $dirconf |
|  |
|  |
| Ejecutables: $BINDIR |
|  |
| Archivos: |
| $dirbin |
|  |
|  |
| Archivos maestros: $MAEDIR |
|  |
| Archivos: |
| $dirmae |
|  |
|  |
| Directorio de arribo de archivos externos: $ARRIDIR |
|  |
| Archivos externos aceptados: $ACEPDIR |
|  |
| Archivos externos rechazados: $RECHDIR |
|  |
| Archivos procesados: $PROCDIR |
|  |
| Reportes de salida: $REPODIR |
|  |
| Logs de auditoría del Sistema: $LOGDIR/InicioX$LOGEXT |
|  |
| Estado del Sistema: INICIALIZADO |
|  |
| Demonio corriendo bajo el no.: <$procssid> " |
|  |
| echo "$mensaje" |
| grabarLog "I" "$mensaje" |
|  |
| } |
|  |
|  |
| #Funcion principal |
|  |
| function main { |
|  |
| variables=(GRUPO BINDIR MAEDIR ARRIDIR ACEPDIR RECHDIR PROCDIR REPODIR LOGDIR LOGEXT LOGSIZE DATASIZE) |
|  |
| comandos=(InicioX.sh DetectaX.sh Interprete.sh ReporteX.pl MoverX.sh StartX.sh StopX.sh GlogX.sh VlogX.sh) |
| CONFDIR=../conf |
|  |
| confFile=InstalarX.conf |
|  |
| chequearVarAmbiente |
|  |
| chequearPaths |
| chequearComandos |
|  |
| grabarLog "I" "Inicio de ejecucion" |
|  |
| chequearMaestros |
| chequearTablas |
|  |
| ingresarCantLoop |
| ingresartEspera |
|  |
| lanzarDetectaX |
|  |
| if [ $? == 1 ]; then |
|  |
| msj="Usted ha elegido no arrancar DetectaX, para hacerlo manualmente debe hacerlo de la siguiente manera: \n |
|  |
| Uso: DetectaX.sh CANTLOOP TESPERA \n |
|  |
| CANTLOOP es la cantidad de ciclos (debe ser un numero entero positivo) que quiere que ejecute el demonio, \n |
| y TESPERA es el tiempo (mayor a 1 minuto) de espera entre cada ciclo.\n" |
|  |
| echo -e $msj |
| grabarLog "I" "$msj" |
|  |
| else |
| #source "$BINDIR/StartX.sh"; |
| StartX.sh "InicioX" "DetectaX.sh $CANLOOP $TESPERA" |
| procssid=$(ps | grep "DetectaX" | cut -f1 -d' ') |
| fi |
|  |
| mostrarMensajeInstalacionFinalizada |
| } |
|  |
| main |

**4.3 - DetectarX.sh**

**Título:** DetectarX.sh

**Justificación**

Es el comando que se encarga de detectar la llegada de archivos al directorio de arribos, validar su formato y nombre, y enviarlos al directorio de archivos aceptados, para luego llamar al Intérprete.sh en caso de que este ultimo directorio no esté vacio. DetectaX.sh es un demonio.

**Archivos de Input, Intermedios y de Output**

Archivos de input

Archivos en directorio de arribos($ARRIDIR), archivo maestro ps-mae.

Archivos de output

Archivos en directorio de aceptados($ACEPDIR), archivos en directorio de rechazados($RECHDIR).

**Parámetros y Opciones**

$CANLOOP (cantidad de veces que corre DetectaX.sh) y $TESPERA (tiempo entre ciclos)

**Invocación Manual y/o automática con ejemplos**

- DetectaX.sh $CANLOOP $TESPERA &

- StartX.sh "comando invocante" "DetectaX.sh $CANLOOP $TESPERA"

**Hipótesis y Aclaraciones especificas del comando**

* Los archivos que tengan nombre y formato válido y estén vacíos se consideran válidos también.
* El DetectarX.sh utiliza un script llamado valPais.sh el cual se detalla a continuación.

Validar archivos que arriban:

* Se considera un formato válido de archivo si tal es “ASCII text” o “empty”.
* Se valida que el formato de código de país sea una letra mayúscula.
* Se valida que el formato de código de sistema sea un número del 0 al 9.
* Se valida que el formato del año sea un número de 4 cifras.
* Se valida que el formato del mes sea un número de 2 cifras.
* Se valida que el período año-mes esté entre 2000-01 y la fecha actual. Esto a su vez automáticamente valida que el año esté entre 2000 y el año actual.
* Se valida que el mes este entre 0 y 12.
* Se utiliza la función externa “validar”(ver 4.3.6) para validar que en el archivo maestro de países y sistemas p-s.mae exista el país, el sistema, y la combinación de ambos, grabando al log un mensaje distinto para cada caso.

Enviar archivo al directorio correspondiente

* Si el archivo pasa la validación, se envía a $ACEPDIR, grabando el log con mensaje de éxito.
* Si el archivo no pasa la validación, se envía a $RECHDIR grabando en el log el mensaje de rechazo.

Ejecutar Intérprete

* Si se encuentran archivos en el directorio $ACEPDIR, se intenta ejecutar el Intérprete
* Si el Intérprete ya estaba corriendo, se escribe en el log un mensaje indicando el error y el número de proceso del Intérprete en ejecución.
* Si el Intérprete no estaba corriendo, se ejecuta.

Tiempo de espera

* Luego de finalizar su tarea, el DetectaX.sh procede a dormir un tiempo de $TESPERA minutos antes de volver a ejecutarse.
* La ejecución del DetectaX.sh finalizará al ciclar $CANLOOP veces.

Uso del GlogX

* Para grabar mensajes de error o de éxito en el log que corresponde al DetectaX.sh, se utiliza la función GlogX

Uso del MoverX

* Para mover archivos del directorio de arribos al de rechazados o aceptados, se utiliza la función MoverX

**Código fuente**

|  |
| --- |
| #!/bin/bash |
|  |
| # |
| # Demonio que detecta llegada de archivos a $ARRIDIR, y los acepta o rechaza |
| # |
| # |
| #ACTUAL="$PWD/Detectar" |
| #ARRIDIR="$ACTUAL/arridir" |
| #ACEPDIR="$ACTUAL/acepdir" |
| #RECHDIR="$ACTUAL/rechdir" |
| #MAEDIR="$ACTUAL/maedir" |
|  |
| source "valPais.sh"; |
| source "$BINDIR/GlogX.sh"; |
| source "$BINDIR/MoverX.sh"; |
| #source "$BINDIR/StartX.sh"; |
| archMae="$MAEDIR/p-s.mae" |
|  |
| #chequeo que me esten pasando 2 parametros |
|  |
| if [[ $# != 2 ]] |
| then |
| echo "Error: Falta un parámetro - debe pasarse como parametro la cantidad de ciclos y el tiempo de espera en segundos" |
| exit 0 |
| fi |
|  |
| mesActual=$(date +%m) |
| anioActual=$(date +%Y) |
| periodoActual=$anioActual$mesActual |
|  |
|  |
| canloop=$1 |
| tespera=$2 |
|  |
| ########## comienza a correr |
| while [[ "$canloop" != 0 ]] |
| do |
| #Grabar en el Log Ciclo Nro "$3-$canloop+1"(Glog) |
|  |
| if [[ -n $(ls $ARRIDIR) ]]  #Chequeo si hay archivos en $ARRIDIR |
| Then |
|  |
|  |
| for arch in "$ARRIDIR"/\*   #para todos los archivos en el directorio $ARRIDIR |
| Do |
| tipoArch="$(file -b "$arch")" #me da el tipo de archivo |
| tipoArch=$(echo $tipoArch | cut -f 1 -d",") |
| valido=true; |
| if [[ "$tipoArch" = "ASCII text" ]] || [[ "$tipoArch" = "empty" ]] #si el archivo es de texto o vacio, es valido |
| then |
|  |
| aux=${arch##\*/} #saco hasta la ultima barra |
| sinExt=${aux%'.'\*} #saco la extension al nombre del archivo |
| ######################### |
| pais=$(echo "$sinExt" | cut -s -f1 -d'-') |
| sistema=$(echo "$sinExt" | cut -s -f2 -d'-') |
| anio=$(echo "$sinExt" | cut -s -f3 -d'-') |
| cantDigAnio=$(echo -n "$anio" | wc -m) |
| mes=$(echo "$sinExt" | cut -s -f4 -d'-') |
| cantDigMes=$(echo -n "$mes" | wc -m) |
| if [[ "$pais" != [A-Z] ]] || [[ "$sistema" != [0-9] ]] || [[ "$cantDigAnio" != 4 ]] || [[ "$cantDigMes" != 2 ]] |
| Then |
| GlogX "DetectaX.sh" "E" "nombre de archivo con formato invalido. Ejemplo de formato valido: A-6-2010-02" "DetectaX" |
| MoverX  "$arch" "$RECHDIR" "DetectaX.sh" |
| valido=false |
| Continue |
| Fi |
| ################################3 |
| periodo=$anio$mes |
| if [[ "$periodo" > "$periodoActual" ]] || [[ "$periodo" < "200000" ]] #2000+00 |
| Then |
| GlogX "DetectaX.sh" "E" "periodo invalido. Debe ser desde 2001-01 hasta $anioActual-$mesActual" "DetectaX" |
| MoverX  "$arch" "$RECHDIR" "DetectaX.sh" |
| valido=false |
| continue |
| fi |
| ########### pais/sistema/mes |
| if [ "$mes" -lt 0 ] || [ "$mes" -gt 12 ] |
| Then |
| GlogX "DetectaX.sh" "E" "Mes invalido" "DetectaX" |
| MoverX  "$arch" "$RECHDIR" "DetectaX.sh" |
| valido=false |
| Continue |
| Fi |
|  |
| validar "$pais" "$sistema" "$archMae" |
| ret=$? |
| if [[ $ret -ne 0 ]] |
| then |
| MoverX  "$arch" "$RECHDIR" "DetectaX.sh" |
| valido=false |
| Continue |
| Fi |
|  |
|  |
| #######aca terminan los invalidos, si llega es que es valido |
| if [[ "$valido" = true ]] |
| Then |
| GlogX "DetectaX.sh" "I" "Archivo valido: $arch" "DetectaX" |
| MoverX  "$arch" "$ACEPDIR" "DetectaX.sh" |
| Else |
| echo "error de programacion, no deberia llegar aca invalido" |
| Fi |
|  |
|  |
| Else |
| GlogX "DetectaX.sh" "E" "tipo de arch invalido (escribir log)" "DetectaX" |
| MoverX  "$arch" "$RECHDIR" "DetectaX.sh" |
| validez=false |
|  |
| fi      #fin del if que evalua el tipo de archivo |
|  |
| done #fin del for que mueve archivos a la carpeta correspondiente y graba logs |
|  |
| Fi |
| ############### todos los archivos que corresponden fueron movidos a $acepdir o $rechdir |
|  |
| if [[ $(ls -A "$ACEPDIR") ]] #si $ACEPTDIR tiene algun archivo |
| Then |
| #GlogX "DetectaX.sh" "I" "$ACEPDIR: Carpeta con archivos, se ejecutara el interprete si no hay otro corriendo" "DetectaX" |
| procssid=$(ps | grep "Interprete" | cut -f1 -d' ') |
| if [ -z $procssid ]; then |
| GlogX "DetectaX.sh" "E" "El Interprete ya esta corriendo. Número de proceso: $procssid" "DetectaX" |
| Else |
| StartX.sh "DetectaX.sh" "Interprete.sh" |
| Fi |
| Fi |
|  |
| #### termino el ciclo, actualizo variable |
| canloop=$(expr $canloop - 1) |
| if [[ $canloop != 0 ]] |
| Then |
| sleep $tespera |
| Fi |
| Done |

**4.3.1 – valPais.sh**

**Título:** valPais.sh

**Justificación**

Es un script que contiene una función llamada "validar" que se ocupa de validar que un código de país y un código de sistema existan en el p-s.mae tanto por separado como juntos. Es utilizado por el DetectaX.sh.

**Archivos de Input, Intermedios y de Output**

Archivos input:

p-s.mae.

Archivos Output:

No tiene.

**Parámetros y Opciones**

$CODIGOPAIS (código del país a validar), $CODIGOSISTEMA (código del sistema a validar), $ARCHMAE (archivo que contiene las líneas donde se encuentran todos los códigos de país y sistema relacionados, concretamente el p-s.mae)

**Invocación Manual y/o automática con ejemplos**

Este script contiene una función aislada, por ende no debería ser ejecutado manualmente. Para hacer uso de la función “validar”, se debe invocar de la siguiente manera:

validar $CODIGOPAIS $CODIGOSISTEMA $ARCHMAE

**Hipótesis y Aclaraciones especificas del comando**

No aplica.

**Código fuente**

|  |
| --- |
| #!/bin/bash |
| function validar() |
| { |
|  |
| if [[ $# != 3 ]] |
| then |
| echo "error en cantidad de parametros, no se pueden validar sistema y pais" |
| exit 20 |
| fi |
|  |
| pais=$1 |
| sistema=$2 |
| archPath=$PWD/$3 |
| paisEncontrado=false |
| sistemaEncontrado=false |
| combinacionEncontrada=false |
|  |
| for i in $(sed "s%^\([A-Z]-[0-9]\)\(-.\*$\)%\1%g" $3) |
| do |
| codPais=$(echo "$i" | cut -s -f1 -d'-' ) |
| codSistema=$(echo "$i" | cut -s -f3 -d'-' ) |
|  |
| if [[ "$pais" = "$codPais" ]] && [[ "$sistema" = "$codSistema" ]] |
| then |
| combinacionEncontrada=true |
| break |
| fi |
|  |
| if [[ "$paisEncontrado" = false ]] && [[ "$pais" = "$codPais" ]] |
| then |
| paisEncontrado=true |
| fi |
|  |
| if [[ "$sistemaEncontrado" = false ]] && [[ "$sistema" = "$codSistema" ]] |
| then |
| sistemaEncontrado=true |
| fi |
|  |
| done |
|  |
| if [[ "$combinacionEncontrada" = true ]] |
| then |
| GlogX "DetectaX.sh" "E" "Valido" "DetectaX" |
| return 0 |
| fi |
|  |
| if [[ "$paisEncontrado" = false ]] && [[ "$sistemaEncontrado" = false ]] |
| then |
| GlogX "DetectaX.sh" "E" "No existe ni sistema ni pais" "DetectaX" |
| return 1 |
| fi |
|  |
| if [[ "$paisEncontrado" = false ]] |
| then |
| GlogX "DetectaX.sh" "E" "No existe pais" "DetectaX" |
| return 2 |
| fi |
|  |
| if [[ "$sistemaEncontrado" = false ]] |
| then |
| GlogX "DetectaX.sh" "E" "No existe sistema" "DetectaX" |
| return 3 |
| fi |
|  |
| if [[ "$paisEncontrado" = true ]] && [[ "$sistemaEncontrado" = true ]] |
| then |
| GlogX "DetectaX.sh" "E" "No existe combinacion" "DetectaX" |
| return 4 |
| fi |
|  |
|  |
| } |
|  |

**4.4 - Interprete.sh**

**Título:** Interprete.sh

**Justificación**

Este comando es el encargado de procesar todos los archivos que lleguen al directorio de ACEPTADOS, dándoles un formato estándar unificado , para luego dejarlos en el directorio de PROCESADOS.

**Archivos de Input, Intermedios y de Output**

Archivos input:

Archivos Recibidos $ACEPDIR/<pais>-<sistema>-<año>-<mes>

Tabla de Separadores $CONFDIR/T1.tab

Tabla de Campos $CONFDIR/T2.tab

Archivos Output:

Archivos de Préstamos Personales por país $PROCDIR/PRESTAMOS.<pais>

Archivos (duplicados) Rechazados $RECHDIR/<nombre del archivo>

Archivos Procesados $PROCDIR/<pais>-<sistema>-<año>-<mes>

Log $LOGDIR/Interprete.$LOGEXT

**Parámetros y Opciones**

No recibe parámetros.

**Invocación Manual y/o automática con ejemplos**

Este comando puede ser invocado tanto manualmente como a través del comando DetectaX. Para su invocación y prueba manual, hay que posicionarse en el directorio bin de la carpeta GRUPO01, en la consola, y ejecutar la función StartX.sh con los parámetros correspondientes. Antes de esto será necesario inicializar el ambiente y mover los archivos de prueba manualmente al directorio ACEPTADOS, ya que no se ejecuta desde el comando DetectaX, que es el encargado de ponerlos en ese directorio automáticamente. Los archivos que son movidos a aceptados, se supone que cumplen con esta condición ya que es el DetectaX el que se encarga de validarlos.

Ejemplo de invocación manual:

StartX.sh ''Invocación manual'' ''Interprete.sh''

**Hipótesis y Aclaraciones especificas del comando**

1- El directorio CONFDIR es conocido.

2- Si el ambiente no se encuentra inicializado correctamente, el interprete no podrá escribir en el log, por lo tanto mostrará el mensaje correspondiente por pantalla y terminará la ejecución.

3- Si en la tabla2.tab no se especifica el campo del cliente o en el archivo este esta vacío, en el registro correspondiente del archivo de procesados.pais el cliente aparece como Cliente sin identificar.

4- Si en la tabla2.tab no se especifica el campo del identificador del cliente o en el archivo este esta vacío, en el registro correspondiente del archivo de procesados.pais el identificador de cliente aparece con un numero negativo(-99999999).

5- Si en la tabla2.tab no se especifica alguno de los campos que intervienen en el cálculo del monto restante o en el archivo alguno está vacío, en el registro correspondiente del archivo de procesados.pais dicho campo tendrá como valor 0 y así no incide en la formula.

6- Todos los archivos poseen el campo de fecha. En caso de que el formato de este sea invalido asignará la fecha genérica(DD/MM/AAAA).

7- Los archivos a procesar son todos válidos excepto los duplicados. Es decir poseen un código de sistema y de pais válido.

8- Si no hay nada en aceptados escribe el mensaje en el log y termina la ejecución.

**Código fuente**

|  |
| --- |
| #!/bin/bash |
| #Definicion de funciones |
|  |
| #Valida si la inicializacion fue hecha correctamente |
| function validarInicio() { |
|  |
| variables=(GRUPO BINDIR MAEDIR ARRIDIR ACEPDIR RECHDIR PROCDIR REPODIR LOGDIR LOGEXT LOGSIZE DATASIZE) |
|  |
| for var in ${variables[\*]} |
| do |
| res=`env | grep $var | cut -d"=" -f 2` |
| if [ -z $res ]; then |
| return 1 |
| fi |
| done |
|  |
| return 0; |
| } |
|  |
| #Verifica que el archivo aceptado no haya sido procesado anteriormente |
| function verificarDuplicado() { |
| if [ ! -f $2/$1 ] |
| then |
| return 0 |
| else |
| return 1 |
| fi |
| } |
|  |
| #Determina el codigo del sistema a traves del nombre del archivo |
| function codigoSystem() { |
| posicionSeparador1=`expr match $1 [Aa-Zz]\*.-` |
| posicionSeparador1=`expr $posicionSeparador1 + 1` |
| posicionSeparador2=`expr match $1 [Aa-Zz]\*.-[0-9]\*.-` |
| longCodigoSistema=`expr $posicionSeparador2 - $posicionSeparador1` |
| codigoSistema=`expr substr $1 $posicionSeparador1 $longCodigoSistema` |
| return $codigoSistema |
| } |
|  |
|  |
|  |
| #INTERPRETE |
|  |
| #Se valida si el ambiente esta inicializado correctamente |
|  |
| CONFDIR=../conf |
|  |
| validarInicio |
| validacion1=$? |
|  |
| if [ $validacion1 -eq 0 ] |
| then |
|  |
| source "$BINDIR/GlogX.sh"; |
| source "$BINDIR/MoverX.sh"; |
|  |
| archivosInput=`ls $ACEPDIR | wc -l` |
|  |
| #Inicializar Log con "Inicio de interprete" y cantidad de archivos de entrada |
| GlogX "Interprete.sh" "I" "Inicio de Interprete" "Interprete" |
|  |
| if [ $archivosInput -ne 0 ] |
| then #Si no hay interprete y el ambiente es el correcto |
|  |
| # archivosInput=`ls $ACEPDIR | wc -l` |
| GlogX "Interprete.sh" "I" "Cantidad de archivos de entrada: $archivosInput" "Interprete" |
|  |
|  |
| #Procesamiento de archivos de $ACEPTDIR(Por cada archivo) |
| registrosInput=0 |
| registrosOutput=0 |
|  |
| for archivo in `ls $ACEPDIR` |
| do |
|  |
| caracter="~" |
| posicion=`expr index $archivo $caracter` |
|  |
| if [ $posicion != 0 ] |
| then |
| rm $ACEPDIR/$archivo |
|  |
| else |
| GlogX "Interprete.sh" "I" "Procesando archivo: $archivo" "Interprete" |
| #Verificar que el archivo no esta duplicado en PROCDIR |
| verificarDuplicado $archivo $PROCDIR |
|  |
| if [ $? -eq 0 ]   #Si no esta duplicado |
| then |
|  |
| #Determinar codigo de pais |
| #echo "ARCHIVO: $archivo" |
| #longCodigoPais=`expr match $archivo [Aa-Zz]\*` |
| #echo "LONGCODPAIS: $longCodigoPais" |
| codigoPais=`expr substr $archivo 1 1` |
|  |
| #Determinar codigo de sistema(Pasarlo a funcion codigoSistema) |
| codigoSystem $archivo |
| codigoSistema=$? |
|  |
| #Determinar separadores |
| Linea=`grep $codigoPais'-'$codigoSistema $CONFDIR/T1.tab` |
|  |
| posicionSepCampos=`expr match $Linea [Aa-Zz]\*.-[0-9]\*.-` |
| posicionSepCampos=`expr $posicionSepCampos + 1` |
| sepCampos=`expr substr $Linea $posicionSepCampos 1` |
|  |
| posicionSepDec=`expr match $Linea [Aa-Zz]\*.-[0-9]\*.-.-` |
| posicionSepDec=`expr $posicionSepDec + 1` |
| sepDecimal=`expr substr $Linea $posicionSepDec 1` |
|  |
| #Determinar campos |
|  |
| #Leer registro(Por cada registro) |
| chmod 777 $PROCDIR |
| touch auxiliar |
|  |
| while IFS=$'\n' read -r linea || [[ -n "$linea" ]] |
| do |
|  |
| echo $linea > auxiliar |
|  |
| #interpretar fecha |
| lineaFecha=`grep $codigoPais'-'$codigoSistema'-CTB\_FE' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| formatoFecha=`grep $codigoPais'-'$codigoSistema'-CTB\_FE' $CONFDIR/T2.tab | cut -f 5 -d"-"` |
| formatoFecha=${formatoFecha%%'.'\*} |
| fecha=`cut -f $lineaFecha -d$sepCampos auxiliar` |
|  |
| if [ "$formatoFecha" = "ddmmyy8" ] |
| then |
| dia=`expr substr $fecha 1 2` |
| mes=`expr substr $fecha 3 2` |
| anio=`expr substr $fecha 5 4` |
| else |
| if [ "$formatoFecha" = "ddmmyy10" ] |
| then |
| dia=`expr substr $fecha 1 2` |
| mes=`expr substr $fecha 4 2` |
| anio=`expr substr $fecha 7 4` |
| else |
| if [ "$formatoFecha" = "yymmdd8" ] |
| then |
| anio=`expr substr $fecha 1 4` |
| mes=`expr substr $fecha 5 2` |
| dia=`expr substr $fecha 7 2` |
| else |
| if [ "$formatoFecha" = "yymmdd10" ] |
| then |
| anio=`expr substr $fecha 1 4` |
| mes=`expr substr $fecha 6 2` |
| dia=`expr substr $fecha 9 2` |
| else |
| GlogX "Interprete.sh" "E" "Formato de fecha invalido" "Interprete" |
| anio="AAAA" |
| mes="MM" |
| dia="DD" |
| fi |
| fi |
| fi |
| fi |
|  |
| #interpretar estado |
| lineaEstado=`grep $codigoPais'-'$codigoSistema'-CTB\_ESTADO' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaEstado ] |
| then |
| estado="Estado sin especificar" |
| else |
| estado=`cut -f $lineaEstado -d$sepCampos auxiliar` |
| if [ -z $estado ] |
| then |
| estado="Estado sin especificar" |
| fi |
| fi |
|  |
| #interpretar codigo prestamo |
| lineaCodPres=`grep $codigoPais'-'$codigoSistema'-PRES\_ID' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaCodPres ] |
| then |
| PRES\_ID=0 |
| else |
| PRES\_ID=`cut -f $lineaCodPres -d$sepCampos auxiliar` |
| if [ -z $PRES\_ID ] |
| then |
| PRES\_ID=0 |
| fi |
| fi |
|  |
| #interpretar monto del prestamo |
| lineaMT\_pres=`grep $codigoPais'-'$codigoSistema'-MT\_PRES' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| formatoNumerico=`grep $codigoPais'-'$codigoSistema'-MT\_PRES' $CONFDIR/T2.tab | cut -f 5 -d"-"` |
| MT\_PRES=`cut -f $lineaMT\_pres -d$sepCampos auxiliar | tr -s $sepDecimal "."` |
| if [ -z $lineaMT\_pres ] |
| then |
| MT\_PRES=0 |
| else |
| MT\_PRES=`cut -f $lineaMT\_pres -d$sepCampos auxiliar | tr -s $sepDecimal "."` |
| if [ -z $MT\_PRES ] |
| then |
| MT\_PRES=0 |
| fi |
| fi |
| MT\_PRES=`echo $MT\_PRES | sed 's/\r$//g'` |
|  |
| #interpretar monto impago |
| lineaMTimp=`grep $codigoPais'-'$codigoSistema'-MT\_IMPAGO' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaMTimp ] |
| then |
| MT\_IMP=0 |
| else |
| MT\_IMP=`cut -f $lineaMTimp -d$sepCampos auxiliar | tr -s $sepDecimal "."` |
| if [ -z $MT\_IMP ] |
| then |
| MT\_IMP=0 |
| fi |
| fi |
| MT\_IMP=`echo $MT\_IMP | sed 's/\r$//g'` |
|  |
| #interpretar monto intereses devengados |
| lineaMT\_inde=`grep $codigoPais'-'$codigoSistema'-MT\_INDE' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaMT\_inde ] |
| then |
| MT\_INDE=0 |
| else |
| MT\_INDE=`cut -f $lineaMT\_inde -d$sepCampos auxiliar | tr -s $sepDecimal "."` |
|  |
| if [ -z $MT\_INDE ] |
| then |
| MT\_INDE=0 |
| fi |
| fi |
| MT\_INDE=`echo $MT\_INDE | sed 's/\r$//g'` |
|  |
| #interpretar monto intereses no devengados |
| linea\_innode=`grep $codigoPais'-'$codigoSistema'-MT\_INNODE' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $linea\_innode ] |
| then |
| MT\_INNODE=0 |
| else |
| MT\_INNODE=`cut -f $linea\_innode -d$sepCampos auxiliar | tr -s $sepDecimal "."` |
| if [ -z $MT\_INNODE ] |
| then |
| MT\_INNODE=0 |
| fi |
| fi |
| MT\_INNODE=`echo $MT\_INNODE | sed 's/\r$//g'` |
|  |
| #interpretar monto debitado |
| lineaMT\_deb=`grep $codigoPais'-'$codigoSistema'-MT\_DEB' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaMT\_deb ] |
| then |
| MT\_DEB=0 |
| else |
| MT\_DEB=`cut -f $lineaMT\_deb -d$sepCampos auxiliar | tr -s $sepDecimal "."` |
| if [ -z $MT\_DEB ] |
| then |
| MT\_DEB=0 |
| fi |
| fi |
| MT\_DEB=`echo $MT\_DEB | sed 's/\r$//g'` |
|  |
| #Calcular monto restante |
| MT\_REST=`echo "$MT\_PRES + $MT\_IMP + $MT\_INDE + $MT\_INNODE - $MT\_DEB"| bc` |
|  |
| #interpretar Id cliente |
| lineaID\_cliente=`grep $codigoPais'-'$codigoSistema'-PRES\_CLI\_ID' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaID\_cliente ] |
| then |
| #paso ID negativo al cliente no identificado |
| ID\_cliente="-99999999" |
| GlogX "Interprete.sh" "E" "ID de cliente sin especificar: $PRES\_ID" "Interprete" |
| else |
| ID\_cliente=`cut -f $lineaID\_cliente -d$sepCampos auxiliar` |
| fi |
|  |
| #interpretar CLiente |
| lineaCliente=`grep $codigoPais'-'$codigoSistema'-PRES\_CLI-' $CONFDIR/T2.tab | cut -f 4 -d"-"` |
| if [ -z $lineaCliente ] |
| then |
| cliente="Cliente sin identificacion" |
| GlogX "Interprete.sh" "E" "Cliente sin especificar: $PRES\_ID" "Interprete" |
| else |
| cliente=`cut -f $lineaCliente -d$sepCampos auxiliar` |
| fi |
|  |
| #Fecha actual |
| fechaActual=`date +%d/%m/%Y` |
|  |
| #Usuario |
| usuario=`whoami` |
|  |
| #Separacion del monto restante en parte entera y parte decimal |
| posicionSeparador=`expr index $MT\_REST '.'` |
| if [ -z $posicionSeparador ]||[ $posicionSeparador -eq 0 ] |
| then |
| MT\_REST\_ENT=$MT\_REST |
| MT\_REST\_DEC=0 |
| else |
| if [ $posicionSeparador -eq 1 ] |
| then |
| MT\_REST\_ENT=0 |
| posicionSeparador=`expr $posicionSeparador + 1` |
| MT\_REST\_DEC=`expr substr $MT\_REST $posicionSeparador 2` |
| else |
| posicionSeparador=`expr $posicionSeparador + 1` |
| MT\_REST\_DEC=`expr substr $MT\_REST $posicionSeparador 2` |
| posicionSeparador=`expr $posicionSeparador - 2` |
| MT\_REST\_ENT=`expr substr $MT\_REST 1 $posicionSeparador` |
| fi |
| fi |
|  |
| #Darle formato a los registros y guardar los correspondientes |
| if [ $MT\_REST\_ENT -le 0 ]&&[ $MT\_REST\_DEC -le 0 ] |
| then |
| GlogX "Interprete.sh" "I" "Prestamo $PRES\_ID cancelado" "Interprete" |
| else |
| registro=`echo "$codigoSistema;$anio;$mes;$dia;$estado;$PRES\_ID;$MT\_PRES;$MT\_IMP;$MT\_INDE;$MT\_INNODE;$MT\_DEB;$MT\_REST;$ID\_cliente;"$cliente";$fechaActual;$usuario"` |
|  |
| linePais=`grep $codigoPais $MAEDIR/p-s.mae` |
| pais=`echo $linePais | cut -f 2 -d"-"` |
| echo $registro >> $PROCDIR/prestamos.$pais |
| registrosOutput=`expr $registrosOutput + 1` |
| fi |
| registrosInput=`expr $registrosInput + 1` |
|  |
| done < $ACEPDIR/$archivo |
|  |
| rm auxiliar |
|  |
| #Grabar en el log la cantidad de registros que entraron y la que salieron |
| GlogX "Interprete.sh" "I" "Registros de Input: $registrosInput" "Interprete" |
| GlogX "Interprete.sh" "I" "Registros de Output: $registrosOutput" "Interprete" |
| #Mover archivo a $PROCDIR |
| MoverX "$ACEPDIR/$archivo" "$PROCDIR" "Interprete.sh" |
|  |
| else  #En caso de estar |
| GlogX "Interprete.sh" "I" "El archivo se encuentra duplicado" "Interprete" |
| MoverX "$ACEPDIR/$archivo" "$RECHDIR" "Interprete.sh" |
|  |
| fi |
| fi |
| done |
| #si archivosInput es igual a cero |
| else |
| GlogX "Interprete.sh" "I" "No hay archivos de entrada" "Interprete" |
| fi |
| #Grabar en el Log fin de interprete |
| GlogX "Interprete.sh" "I" "Interprete finalizado" "Interprete" |
| #si el ambiente no esta inicializado |
| else |
| echo "Interprete.sh - El ambiente no esta inicializado" |
| echo "Interprete.sh - No se ejecuta el comando" |
| fi |

**4.5 - ReporteX.pl**

**Título:** ReporteX.pl

**Justificación**

El propósito de este comando es resolver, mostrar y, eventualmente, grabar consultas vinculadas a la comparación entre dos fuentes: el archivo de préstamos por país (prestamos.<país>) y el archivo maestro (PPI.mae).

Se pueden realizar tres tipos de reportes: de recomendación de recálculo, de diferencia de montos y de diferencia de porcentajes.

* **Reporte de recomendación para recálculo:** El primero determina el valor de la “recomendación” que puede ser “RECALCULO” o “BUENO”. La recomendación se determina de la siguiente manera: si estado contable del maestro es igual a “SMOR” y el estado contable prestamos.pais es distinto a “SMOR”, entonces la recomendación es “RECALCULO”. Si monto restante del maestro menor a monto restante de prestamos.pais, recomendación es “RECALCULO”.
* **Reporte de diferencia de montos:** Para esta consulta se debe solicitar que se ingrese un monto de comparación X. Esta consulta responde a la pregunta: En qué casos la diferencia (en valor absoluto) entre el monto restante del maestro y el monto restante del país es mayor a $ X.
* **Reporte de diferencia de porcentajes:** Para esta consulta se debe solicitar que se ingrese un monto de comparación X. Esta consulta responde a la pregunta: En qué casos la diferencia (en valor absoluto) entre el monto restante del maestro y el monto restante del país es mayor a X %.

**Archivos de Input, Intermedios y de Output**

Archivos input:

* Maestro de Países y Sistemas $MAEDIR/p-s.mae .
* Maestro Contable de Préstamos Personales Impagos $MAEDIR/PPI.mae .
* Préstamos Personales por país $PROCDIR/PRESTAMOS.<pais> .

Archivos Output:

* Archivos de Recalculo $REPODIR/RECALCULO.
* Listados $REPODIR/ReporteX.xxx .

**Parámetros y Opciones**

Al comando se le debe pasar obligatoriamente uno de los siguientes parámetros, correspondientes a cada uno de los reportes anteriormente explicados:

* + -cr: lanza la comparación para recálculo.
  + -dp: lanza la comparación para diferencia porcentual.
  + -dm: lanza la comparación para diferencia porcentual.

A su vez, este parámetro debe ir acompañado con la especificación del país acerca del cual se desean realizar los reportes. El siguiente parámetro es obligatorio, y común a todos los subcomandos:

* + - **-p**=<país>: Es el país acerca del cual quiero obtener la información.

Una vez ingresados estos dos parámetros ya se puede ejecutar el comando, pero además cada uno posee algunos filtros, para especificar la consulta. Las siguientes opciones son comunes a todos los parámetros:

* + - **-s**=<sistema>: indica el código de sistema al cual quiere que pertenezcan los registros a mostrar.
    - **-a**=<año>: indica el año contable al cual quiere que pertenezcan los registros a mostrar. Si se ingresa este parámetro, no se puede ingresar ni un período ni un rango de períodos.
    - **-pe**=<período>=[AAAA/MM]: indica el mes de un año en el cual se quiere que esté comprendido el mes contable del reporte. Si se ingresa este parámetro no se puede filtrar por año ni por rango de períodos.
    - **-rp**=<rango de períodos>=[AAAA/MM]-[AAAA/MM]: ídem punto anterior para un rango de meses. Si se ingresa este parámetro no se puede filtrar por año ni por período.
    - **-g**: guarda la vista del reporte mostrado por pantalla en un archivo de texto.

Finalmente, sólo para los reportes de diferencia existe la siguiente opción:

* + - **-x**=<valor>: indica la diferencia mínima que se desea mostrar (en valor absoluto) en cuanto a monto o porcentaje en estos reportes.

**Invocación Manual y/o automática con ejemplos**

* Para reporte de recomendación para recálculo:
  + - ./Reporte.pl -cr -p=Argentina -a=2013
    - ./Reporte.pl -cr -p=Argentina -pe=2013/12
    - ./Reporte.pl -cr -p=Argentina -a=2013 -s=7
* Para reporte de diferencias:
  + - ./Reporte.pl -dp -p=Argentina -a=2013 -x=5
    - ./Reporte.pl -dm -p=Argentina -rp=2011/12-2013/01 s=6
    - ./Reporte.pl -dp -p=Argentina -a=2013 -s=7 -x=50 -g

**Hipótesis y Aclaraciones especificas del comando**

* No se validan los rangos. Si las opciones ingresadas son inválidas, se retornan reportes vacíos.
* Se supuso que si en los reportes por diferencia no se ingresa un valor de X, se pasa X=0.

**Código fuente**

|  |
| --- |
| #!/usr/bin/perl |
|  |
| ######################## |
| #                      # |
| #  VARIABLES GLOBALES  # |
| #                      # |
| ######################## |
|  |
| $MAEDIR="";             # Directorio de archivos maestros |
| $PROCDIR="";    # Directorio de archivos para procesamiento |
| $REPODIR="";    # Directorio donde se guardan los reportes |
|  |
| $PAIS\_ID="";    # id del pais |
| $PAIS\_DESC="";  # descripcion del pais |
| $SIS\_ID="";             # id del sistema |
| $CTB\_ANIO="";   # año contable |
| $CTB\_MES="";    # mes contable |
| $PRES\_ID="";    # id del préstamo |
|  |
| $MT\_PRES=0;     # monto del préstamo |
| $MT\_IMPAGO=0;   # monto impago |
| $MT\_INDE=0;     # monto intereses devengados |
| $MT\_INNODE=0;   # monto intereses no devengados |
| $MT\_DEB=0;      # monto debitado |
| $MT\_REST=0;     # monto restante |
| $GUARDAR\_REPORTE=0; # flag para saber si guardar el reporte |
| $VALOR\_DIFERENCIA=0;# diferencia para los comandos de diferencia |
|  |
| $PERIODO=""; |
| $RANGO\_PERIODOS=""; |
| $COMANDOS\_USADOS=""; |
| $USUARIO=""; |
|  |
| %registros\_ppi; |
| %registros\_prestamos; |
|  |
| @reg\_ppi; |
| @reg\_p\_p; |
|  |
| @reporte; |
|  |
| ################ |
| #              # |
| #  SUBRUTINAS  # |
| #              # |
| ################ |
|  |
| # |
| # Inicializo las variables globales |
| # |
| sub inicializarGlobales{ |
|  |
| $PAIS\_ID="";    # id del pais |
| $PAIS\_DESC="";  # descripcion del pais |
| $SIS\_ID="";     # id del sistema |
| $CTB\_ANIO="";   # año contable |
| $CTB\_MES="";    # mes contable |
| $PRES\_ID="";    # id del préstamo |
|  |
| $MT\_PRES=0;     # monto del préstamo |
| $MT\_IMPAGO=0;   # monto impago |
| $MT\_INDE=0;     # monto intereses devengados |
| $MT\_INNODE=0;   # monto intereses no devengados |
| $MT\_DEB=0;      # monto debitado |
| $MT\_REST=0;     # monto restante |
| $GUARDAR\_REPORTE=0; # flag para saber si guardar el reporte |
| $VALOR\_DIFERENCIA=0; |
|  |
| $PERIODO=""; |
| $RANGO\_PERIODOS=""; |
| $COMANDOS\_USADOS=""; |
|  |
| %registros\_ppi=(); |
| %registros\_prestamos=(); |
|  |
| @reporte=(); |
| @reg\_ppi=(); |
| @reg\_p\_p=(); |
| } |
|  |
| # |
| # Se fija si la respuesta es si o no, o si es invalida |
| # |
| sub normalizarValorDeUnDigito{ |
|  |
| my $valor = @\_[0]; |
|  |
| if( int($valor) < 10 ) |
| { |
| $dia = '0'.int($valor); |
| } |
|  |
| return $valor; |
| } |
|  |
| # |
| # Se fija si la respuesta es si o no, o si es invalida |
| # |
| sub validarRespuesta{ |
|  |
| my $rta = @\_[0]; |
|  |
| if ( uc($rta) eq uc("n") ) |
| { |
| return(0); |
| last; |
| } |
| elsif ( uc($rta) eq uc("s") ) |
| { |
| return(1); |
| } |
| else |
| { |
| print "¡Respuesta incorrecta!"; |
| return(-1); |
| } |
| } |
|  |
| # |
| # Confirma si sigue la ejecucion del comando actual |
| # |
| sub confirma{ |
|  |
| my $mensaje = @\_[0]; |
| my $respuesta = -1; |
| while ( $respuesta == -1 ) |
| { |
| print $mensaje; |
| $rta = <STDIN>; |
| chomp($rta); |
|  |
| $respuesta = &validarRespuesta( $rta ); |
| } |
| return $respuesta; |
| } |
|  |
| sub buscarDatosPais{ |
|  |
| my @valores\_registro; |
|  |
| # obtengo los valores del archivo de paises y  sistemas |
| open (P\_S,"$MAEDIR/p-s.mae"); |
|  |
| # busco el codigo del pais ingresado |
| while (<P\_S>) |
| { |
| @valores\_registro = split('-',$\_); |
|  |
| if( uc(@valores\_registro[1]) eq uc($PAIS\_DESC)  ) |
| { |
| $PAIS\_ID = @valores\_registro[0]; |
| $PAIS\_DESC = @valores\_registro[1]; |
| last; |
| } |
| } |
| } |
|  |
| # |
| # Obtiene los directorios de los archivos maestros y de procesamiento, |
| # y donde se guardaran los reportes. |
| # |
| sub obtenerDirectorios{ |
|  |
| my @valores\_registro; |
| my $cantDir = 0; |
|  |
| # Abro el archivo InstalarX.conf |
| open(CONF,"../conf/InstalarX.conf") || die "ERROR: No puedo abrir el fichero InstalarX.conf\n"; |
|  |
| # obtengo el maedir, procdir y repodir |
| while (<CONF>) |
| { |
| # quito eol |
| chomp; |
|  |
| @valores\_registro = split("=",$\_); |
| if (@valores\_registro[0] eq "MAEDIR") |
| { |
| $MAEDIR = @valores\_registro[1]; |
| $USUARIO = @valores\_registro[2]; |
| $cantDir++; |
| } |
| elsif (@valores\_registro[0] eq "PROCDIR") |
| { |
| $PROCDIR = @valores\_registro[1]; |
| $USUARIO = @valores\_registro[2]; |
| $cantDir++; |
| } |
| elsif (@valores\_registro[0] eq "REPODIR") |
| { |
| $REPODIR = @valores\_registro[1]; |
| $USUARIO = @valores\_registro[2]; |
| $cantDir++; |
| } |
| if ($cantDir == 3) |
| { |
| last; |
| } |
| } |
| close(CONF); |
|  |
| #print "DIRS: MAE: $MAEDIR, PROC: $PROCDIR, REPO: $REPODIR\n\n"; |
| } |
|  |
| # |
| # Graba el reporte en un archivo de texto. |
| # param1 : nombre del archivo donde grabar. |
| # param2 : cantidad de columnas que tiene el reporte. |
| # |
| sub grabarRecalculo{ |
|  |
| my $nombreArchivo = @\_[0]; |
| my $CodigoSistema; |
| my $AnioContable; |
| my $MesContable; |
| my $DiaContable; |
| my $EstadoContable; |
| my $CodigoPrestamo; |
| my $MontoPrestamo; |
| my $MontoImpago; |
| my $MontoInteresDevengado; |
| my $MontoInteresNoDevengado; |
| my $MontoDebitado; |
| my $MontoRestante; |
| my $CodigoCliente; |
| my $NombreCliente; |
| my $FechaGrabacion; |
| my $UsuarioGrabacion; |
| my $linea=""; |
|  |
| my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time); |
| $year += 1900; |
| $mon++; |
|  |
| $mon = "".&normalizarValorDeUnDigito($mon); |
| $mday = "".&normalizarValorDeUnDigito($mday); |
|  |
| open(ARCHIVO\_REPORTE,">>$REPODIR/$nombreArchivo") || die "ERROR: No puedo abrir el fichero $nombreArchivo\n"; |
|  |
| # recorro los datos y voy viendo si es necesario el recalculo |
| foreach my $llave (keys %registros\_ppi) |
| { |
| # me fijo que exista en ambos archivos |
| if ( exists ($registros\_prestamos{$llave}) ) |
| { |
| # obtengo los datos del registro |
| @reg\_p\_p = split(";",$registros\_prestamos{$llave}); |
| @reg\_ppi = split(";",$registros\_ppi{$llave}); |
|  |
| #print "reg\_p\_p: $registros\_prestamos{$llave}\n"; |
| #print "reg\_ppi: $registros\_ppi{$llave}\n"; |
|  |
| # si es recalculo lo grabo |
| if ( ( $reg\_ppi[5] eq "SMOR" && $reg\_p\_p[4] ne "SMOR" ) || ( $reg\_ppi[14] lt $reg\_p\_p[11]  ) ) |
| { |
| $CodigoSistema = @reg\_ppi[1];chomp($CodigoSistema); |
| $AnioContable = @reg\_ppi[2];chomp($AnioContable); |
| $MesContable = @reg\_ppi[3];chomp($MesContable); |
| $DiaContable = @reg\_p\_p[3];chomp($EstadoContable); |
| $EstadoContable = @reg\_ppi[5];chomp($EstadoContable); |
| $CodigoPrestamo = @reg\_ppi[7];chomp($CodigoPrestamo); |
| $MontoPrestamo = @reg\_ppi[9];chomp($MontoPrestamo); |
| $MontoImpago = @reg\_ppi[10];chomp($MontoImpago); |
| $MontoInteresDevengado = @reg\_ppi[11];chomp($MontoInteresDevengado); |
| $MontoInteresNoDevengado = @reg\_ppi[12];chomp($MontoInteresNoDevengado); |
| $MontoDebitado = @reg\_ppi[13];chomp($MontoDebitado); |
| $MontoRestante = @reg\_ppi[14];chomp($MontoRestante); |
| $CodigoCliente = @reg\_p\_p[12];chomp($CodigoCliente); |
| $NombreCliente = @reg\_p\_p[13];chomp($NombreCliente); |
| $FechaGrabacion = "$year/$mon/$mday"; |
| $UsuarioGrabacion = $USUARIO;chomp($UsuarioGrabacion); |
|  |
| # grabo el registro |
| #$linea = $CodigoSistema.";".$AnioContable.";".$MesContable.";".$DiaContable.";".$EstadoContable.";".$CodigoPrestamo.";".$MontoPrestamo.";".$MontoImpago.";".$MontoInteresDevengado.";".$MontoInteresNoDevengado.";".$MontoDebitado.";".$MontoRestante.";".$CodigoCliente.";".$NombreCliente.";".$FechaGrabacion.";".$UsuarioGrabacion."\n"; |
| print ARCHIVO\_REPORTE "$CodigoSistema;$AnioContable;$MesContable;$DiaContable;$EstadoContable;$CodigoPrestamo;$MontoPrestamo;"; |
| print ARCHIVO\_REPORTE "$MontoImpago;$MontoInteresDevengado;$MontoInteresNoDevengado;$MontoDebitado;$MontoRestante;$CodigoCliente;"; |
| print ARCHIVO\_REPORTE "$NombreCliente;$FechaGrabacion;$UsuarioGrabacion\n"; |
|  |
| } |
| } |
| } |
|  |
| close(ARCHIVO\_REPORTE); |
| } |
|  |
| # |
| # Ayuda |
| # |
| sub mostrarAyuda{ |
|  |
| # obtengo el comando del cual quiere obtener ayuda. |
| my $param = @\_[0]; |
| my $parametros\_obligatorios = "Parámetros obligatorios:\n"."                -p=<país>: Es el país acerca del cual quiero obtener la información.\n\n"."Parámetros para otros filtros:\n"."          -x=<valor de diferencia>\n"."           -s=<sistema>\n"."               -a=<año>\n"."           -pe=<período=[AAAA/MM]>: indica el mes de un año en el cual se quiere que esté\n"."                     comprendido el mes contable del reporte.\n"."           -rp=<rango de períodos=[AAAA/MM]-[AAAA/MM]>: ídem punto anterior\n"."               para un rango de meses.\n\n"; |
| chomp($param); |
|  |
| if ( uc($param) eq "" ) |
| { |
| print "Ayuda del Generador de reportes del Grupo 1.\n"; |
| print "Modo de uso:     ReporteX <subcomando> [opciones][argumenos]\n"; |
| print "Tipee 'ReporteX -a <subcomando>' para obtener información acerca del subcomando.\n\n"; |
| print "Subcomandos disponibles:\n"; |
| print " -a\n"; |
| print " -cr\n"; |
| print " -dm\n"; |
| print " -dp\n"; |
| print " -g\n\n"; |
| } |
| elsif ( uc($param) eq uc("-a") ) |
| { |
| print "Muestra información acerca del parámetro pasado como parámetro.\n"; |
| print "Si se escribe el subcomando sólo, muestra la ayuda general.\n\n"; |
| } |
| elsif ( uc($param) eq uc("-cr") ) |
| { |
| print "Este subcomando genera el reporte de recomendación para recálculo,\n"; |
| print "indicando para cuáles de las cuentas ingresadas es necesario hacer\n"; |
| print "un recálculo de la deuda en la base maestra.\n\n"; |
| print "Parámetros obligatorios:\n"; |
| print "         -p=<país>: Es el país acerca del cual quiero obtener la información.\n\n"; |
| print "Parámetros para otros filtros:\n"; |
| print "         -s=<sistema>\n"; |
| print "         -a=<año>\n"; |
| print "         -pe=<período=[AAAA/MM]>: indica el mes de un año en el cual se quiere que esté\n"; |
| print "                 comprendido el mes contable del reporte.\n"; |
| print "         -rp=<rango de períodos=[AAAA/MM]-[AAAA/MM]>: ídem punto anterior\n"; |
| print "             para un rango de meses.\n\n"; |
| } |
| elsif ( uc($param) eq uc("-dm") ) |
| { |
| print "Este comando genera un reporte con los registros en cuyos casos casos\n"; |
| print "la diferencia (en valor absoluto) entre el monto restante del maestro\n"; |
| print "y el monto restante del país es mayor a X monto. El valor X se pasa como\n"; |
| print "parámetro. En caso de no pasarse nada se muestran todos los registros.\n\n"; |
| print "$parametros\_obligatorios"; |
| } |
| elsif ( uc($param) eq uc("-dp") ) |
| { |
| print "Este comando genera un reporte con los registros en cuyos casos casos\n"; |
| print "la diferencia (en valor absoluto) entre el monto restante del maestro\n"; |
| print "y el monto restante del país es mayor al X %. El valor X se pasa como\n"; |
| print "parámetro. En caso de no pasarse nada se muestran todos los registros.\n\n"; |
| print "$parametros\_obligatorios"; |
| } |
|  |
| elsif ( uc($param) eq uc("-g") ) |
| { |
| print "Si se incluye se guardará el reporte mostrado en pantalla en archivo de texto.\n\n"; |
| } |
| else |
| { |
| print "El comando ingresado no existe para el Generador de reportes.\n\n"; |
| } |
|  |
| } |
|  |
| # |
| # Obtengo los valores del archivo de prestamos por pais |
| # |
| sub obtenerPrestamosPais{ |
|  |
| my @valores\_registro; |
| my $anio\_ctb; |
| my $mes\_ctb; |
| my $dia\_ctb; |
| my $fecha\_grab; |
|  |
| my $clave\_p\_p; |
| my $aux; |
|  |
| # paso a minusculas la descripcion del pais (para abrir el archivo de prestamos por pais) |
| $PAIS\_DESC =~ tr/A-Z/a-z/; |
|  |
| #  el archivo de prestamos.pais |
| open(PRESTAMOS\_PAIS,"$PROCDIR/prestamos.".$PAIS\_DESC); |
|  |
| while(<PRESTAMOS\_PAIS>) |
| { |
| chomp; # quito el eol |
|  |
| # obtengo los datos y los almaceno en un array |
| @valores\_registro = split(';',$\_); |
|  |
| # tomo los datos necesarios para la comparacion |
| $anio\_ctb   = @valores\_registro[1]; |
| $mes\_ctb    = &normalizarValorDeUnDigito(@valores\_registro[2]); |
| $dia\_ctb    = @valores\_registro[3]; |
| $fecha\_grab = @valores\_registro[14]; |
|  |
| # compongo la clave de busqueda |
| $clave\_p\_p  = @valores\_registro[5].int($anio\_ctb).int($mes\_ctb); |
|  |
| # almaceno los valores en una estructura hash |
| if ( exists $registros\_prestamos{$clave\_p\_p} ) |
| { |
| # obtengo los datos almacenados |
| $aux = split(';',$registros\_prestamos{$clave\_p\_p}); |
| # si la fecha de grabacion del nuevo registro es mas actual lo almaceno |
| if ( uc($aux[14]) lt uc($fecha\_grab)  ) |
| { |
| # borro y reinserto datos validos |
| delete($registros\_prestamos{$clave\_p\_p}); |
| $registros\_prestamos{$clave\_p\_p} = $\_; |
|  |
| } |
| # si no, no hago nada... |
| } |
| else |
| { |
| $registros\_prestamos{$clave\_p\_p} = $\_; |
| } |
|  |
| } |
|  |
| close(PRESTAMOS\_PAIS); |
|  |
| #print "REGISTROS PRESTAMOS PAIS\n"; |
| #print map "Hash: $\_ = $registros\_prestamos{$\_}\n", keys %registros\_prestamos; |
| } |
|  |
| # |
| # Obtengo los valores del archivo de prestamos personales impagos |
| # |
| sub obtenerPrestamosImpagos{ |
|  |
| my @valores\_registro; |
| my @aux; |
|  |
| my $reg; |
| my $clave\_ppi; |
|  |
| # abro el archivo maestro (PPI) |
| open (PPI,"$MAEDIR/PPI.mae") || die "ERROR: No puedo abrir el fichero PPI.\n"; |
|  |
| # realizo la lectura para ver que registros coinciden con el filtro |
| while (<PPI>) |
| { |
| chomp; # quito el eol |
|  |
| # guardo los valores en una cadena |
| @valores\_registro = split(';',$\_); |
|  |
| @valores\_registro[3] = &normalizarValorDeUnDigito(@valores\_registro[3]); |
|  |
| # verifico que cumpla todas las condiciones |
| if ( uc($PAIS\_ID) ne uc(@valores\_registro[0])){ next; } |
| if ( $SIS\_ID ne "" ) |
| { |
| if ( uc($SIS\_ID) ne uc(@valores\_registro[1]) ){ next; } |
| } |
| if ( $CTB\_ANIO ne "" ) |
| { |
| if ( uc($CTB\_ANIO) ne uc(@valores\_registro[2])){ next; } |
| } |
| if ( $PERIODO ne "" ) |
| { |
| if ( uc($PERIODO) ne uc(@valores\_registro[2]."/".@valores\_registro[3])){ next; } |
| } |
| if ( $RANGO\_PERIODOS ne "" ) |
| { |
| if ( uc(substr($RANGO\_PERIODOS,0,7)) gt uc(@valores\_registro[2]."/".@valores\_registro[3]) |
| || uc(substr($RANGO\_PERIODOS,8,7)) lt uc(@valores\_registro[2]."/".@valores\_registro[3])){ next; } |
| } |
|  |
| # obtengo el id del prestamo |
| $PRES\_ID = @valores\_registro[7]; |
|  |
| # cambio comas por puntos |
| @valores\_registro[9] =~ s/\,/\./; |
| @valores\_registro[10] =~ s/\,/\./; |
| @valores\_registro[11] =~ s/\,/\./; |
| @valores\_registro[12] =~ s/\,/\./; |
| @valores\_registro[13] =~ s/\,/\./; |
| # obtengo los valores y con ellos hago los calculos |
| $MT\_PRES    = sprintf("%.2f",@valores\_registro[9]); |
| $MT\_IMPAGO  = sprintf("%.2f",@valores\_registro[10]); |
| $MT\_INDE    = sprintf("%.2f",@valores\_registro[11]); |
| $MT\_INNODE  = sprintf("%.2f",@valores\_registro[12]); |
| $MT\_DEB     = sprintf("%.2f",@valores\_registro[13]); |
|  |
| # calculo el monto restante |
| $MT\_REST    = sprintf("%.2f",$MT\_PRES + $MT\_IMPAGO + $MT\_INDE + $MT\_INNODE - $MT\_DEB); |
|  |
| # agrego a los valores el monto restante |
| push(@valores\_registro,$MT\_REST); |
|  |
| print "vañpres registro: @valores\_registro\n"; |
|  |
| # almaceno el array en un string |
| $reg = join(';',@valores\_registro); |
|  |
| print "Registro ppi procesado: $reg\n"; |
|  |
| # creo la clave del registro |
| $clave\_ppi = $PRES\_ID.int(@valores\_registro[2]).int(@valores\_registro[3]); |
|  |
| # guardo los valores obtenidos en una estructura hash |
| if ( ! exists $registros\_ppi{$clave\_ppi} ) |
| { |
| $registros\_ppi{$clave\_ppi} = $reg; |
| } |
| } |
|  |
| # cierro el archivo |
| close(PPI); |
|  |
| print "REGISTROS PPI\n"; |
| print map "Hash: $\_ = $registros\_ppi{$\_}\n", keys %registros\_ppi; |
| } |
|  |
| # |
| # Calculo y muestro la recomendacion |
| # |
| sub mostrarRecomendacion{ |
|  |
| my @aux; |
|  |
| my $linea; |
| my $recomendacion; |
|  |
| # recorro los datos y voy viendo si es necesario el recalculo |
| foreach my $llave (keys %registros\_ppi) |
| { |
| # me fijo que exista en ambos archivos |
| if ( exists ($registros\_prestamos{$llave}) ) |
| { |
| # obtengo los datos del registro |
| @reg\_p\_p = split(";",$registros\_prestamos{$llave}); |
| @reg\_ppi = split(";",$registros\_ppi{$llave}); |
|  |
| #print "reg\_p\_p: $registros\_prestamos{$llave}\n"; |
| #print "reg\_ppi: $registros\_ppi{$llave}\n"; |
|  |
| # defino si hay que hacer recalculo o no |
| if ( ( $reg\_ppi[5] eq "SMOR" && $reg\_p\_p[4] ne "SMOR" ) || ( $reg\_ppi[14] lt $reg\_p\_p[11]  ) ) |
| { |
| $recomendacion = "RECALCULO"; |
| } |
| else |
| { |
| $recomendacion = "BUENO"; |
| } |
| #           prestamo    -   cliente  -  estado cont mae-estado cont pais- mt rest mae   - mt rest pais |
| $linea = $reg\_ppi[7].";".$reg\_p\_p[12].";".$reg\_ppi[5].";".$reg\_p\_p[4].";".$reg\_ppi[14].";".$reg\_p\_p[11].";".$recomendacion; |
|  |
| push (@aux,$linea); |
| } |
| } |
|  |
| # ordeno el reporte alfabéticamente |
| @aux = sort { lc($a) cmp lc($b) } @aux; |
|  |
| # meto el auxiliar en el reporte |
| foreach $linea (@aux) |
| { |
| my @aux2; |
| @aux2 = split(";",$linea); |
| @aux2 = reverse(@aux2); |
| # cambio los puntos por comas |
| @aux2[1] =~ s/\./\,/; |
| @aux2[2] =~ s/\./\,/; |
| push (@reporte,@aux2[0]); # recomendacion |
| push (@reporte,@aux2[1]); # mt rest pais |
| push (@reporte,@aux2[2]); # mt rest mae |
| push (@reporte,@aux2[3]); # estado cont pais |
| push (@reporte,@aux2[4]); # estado cont mae |
| push (@reporte,@aux2[5]); # cliente |
| push (@reporte,@aux2[6]); # prestamo |
| } |
|  |
| # imprimo la cabecera |
| my $titulos = "\nREPORTE DE RECÁLCULOS\n\nParámetros ingresados\n"; |
| $titulos .= $COMANDOS\_USADOS."\n\n"; |
| $titulos .= sprintf         "%-15s%-15s%-15s%-15s%-15s%-15s%-15s%s", |
| "Prestamo","Cliente","ECM","ECP","MRM","MRP","Recomendacion", |
| "\n---------------------------------------------------------------------------------------------------------\n"; |
| print "$titulos"; |
|  |
| # muestro el reporte |
| @reporte = reverse(@reporte); |
| my $cantCol = 0; |
| my $cantReg = 0; |
| my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time); |
| $year += 1900; |
| $mon++; |
| $mon = &normalizarValorDeUnDigito($mon); |
| $mday = &normalizarValorDeUnDigito($mday); |
| $hour = &normalizarValorDeUnDigito($hour); |
| $min = &normalizarValorDeUnDigito($min); |
| $sec = &normalizarValorDeUnDigito($sec); |
|  |
| my $descriptor = $year.$mon.$mday.$hour.$min.$sec; |
|  |
| # si me pidieron guardar el reporte creo el archivo |
| if ( $GUARDAR\_REPORTE ) |
| { |
| open(REPORTEX,">>$REPODIR/ReporteX\_RECALCULO.$descriptor") || die "ERROR: No puedo abrir el fichero ReporteX\_$tipo\_rep.$descriptor\n"; |
| print REPORTEX "$titulos"; |
| } |
|  |
| foreach $elem (@reporte) |
| { |
| chomp; |
| printf ("%-15s",$elem); |
|  |
| # guardo si me lo pidieron |
| if ( $GUARDAR\_REPORTE ) |
| { |
| printf ( REPORTEX "%-15s",$elem); |
| } |
|  |
| $cantCol++; |
|  |
| # paso al siguiente renglon |
| if ( $cantCol == 7 ) |
| { |
| print "\n"; |
| if ( $GUARDAR\_REPORTE ) |
| { |
| print REPORTEX "\n"; |
| } |
| $cantCol = 0; |
| $cantReg++; |
| } |
| } |
|  |
| # imprimo la cantidad de registros calculados |
| print "\n"; |
| printf("%d registros",$cantReg); |
| print "\n"; |
|  |
| if ( $GUARDAR\_REPORTE ) |
| { |
| print REPORTEX "\n"; |
| printf (REPORTEX "%d registros",$cantReg); |
| print REPORTEX "\n"; |
| close(REPORTEX); |
| } |
| } |
|  |
| # |
| # Muestra el reporte de diferencia porcentual o en monto, segun el parametro |
| # |
| sub mostrarDiferencia{ |
|  |
| my @aux; |
|  |
| my $linea; |
| my $recomendacion; |
| my $tipo\_rep; |
|  |
| my $monto\_mae; |
| my $monto\_pais; |
| my $diferencia; |
| my $resultado; |
|  |
| my $cmd = @\_[0]; |
|  |
| if ( $cmd eq "-dp" ) |
| { |
| $tipo\_rep  = "PORCENTAJE"; |
| } |
| else |
| { |
| $tipo\_rep  = "MONTO"; |
| } |
|  |
| # recorro los datos |
| foreach my $llave (keys %registros\_ppi) |
| { |
| # me fijo que exista en ambos archivos |
| if ( exists ($registros\_prestamos{$llave}) ) |
| { |
| # obtengo los datos del registro |
| @reg\_p\_p = split(";",$registros\_prestamos{$llave}); |
| @reg\_ppi = split(";",$registros\_ppi{$llave}); |
|  |
| $reg\_p\_p[11] =~ s/\,/\./; |
| $reg\_p\_p[14] =~ s/\,/\./; |
|  |
| $monto\_pais = sprintf("%.2f",$reg\_p\_p[11]); |
| $monto\_mae = sprintf("%.2f",$reg\_ppi[14]); |
| $diferencia = $monto\_mae - $monto\_pais; |
|  |
| if ( $cmd eq "-dp" ) |
| { |
| $tipo\_rep  = "PORCENTAJE"; |
| $resultado = $diferencia \* 100 / $monto\_mae; |
| $resultado = sprintf("%.2f",$resultado); |
| } |
| else |
| { |
| $tipo\_rep  = "MONTO"; |
| $resultado = sprintf("%.2f",$diferencia); |
| } |
|  |
| # si el monto o porcentaje es mayor al pasado como parametro |
| if ( abs($resultado) >= abs($VALOR\_DIFERENCIA) ) |
| { |
| #           prestamo - mt rest mae   - mt rest pais |
| $linea = $reg\_ppi[7].";".$reg\_ppi[14].";".$reg\_p\_p[11].";".$resultado; |
| push (@aux,$linea); |
| } |
| } |
| } |
|  |
| # ordeno el reporte alfabéticamente |
| @aux = sort { lc($a) cmp lc($b) } @aux; |
| @aux = reverse(@aux); |
|  |
| # meto el auxiliar en el reporte |
| foreach $linea (@aux) |
| { |
| my @aux2; |
| @aux2 = split(";",$linea); |
| @aux2 = reverse(@aux2); |
|  |
| @aux2[1] = sprintf("\$ %s",@aux2[1]); |
| @aux2[2] = sprintf("\$ %s",@aux2[2]); |
|  |
| if ( $cmd eq "-dm" ) |
| { |
| @aux2[0] = sprintf ("\$ %s",@aux2[0] ); |
| } |
| else |
| { |
| @aux2[0] = sprintf ("%s %%",@aux2[0] ); |
| } |
|  |
| @aux2[1] =~ s/\./\,/; |
| @aux2[2] =~ s/\./\,/; |
| push (@reporte,@aux2[0]); # diferencia |
| push (@reporte,@aux2[1]); # mt rest pais |
| push (@reporte,@aux2[2]); # mt rest mae |
| push (@reporte,@aux2[3]); # prestamo |
| } |
|  |
| # imprimo la cabecera |
| my $titulos = "\nREPORTE DE DIFERENCIA EN ".$tipo\_rep."\n\nParámetros ingresados\n"; |
| $titulos .= $COMANDOS\_USADOS."\n\n"; |
| $titulos .= sprintf         "%-15s%-15s%-15s%-15s%s", |
| "Prestamo","MRM","MRP","Diferencia", |
| "\n------------------------------------------------------------\n"; |
| print "$titulos"; |
|  |
| # muestro el reporte |
| @reporte = reverse(@reporte); |
| my $cantCol = 0; |
| my $cantReg = 0; |
| my ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) = localtime(time); |
| $year += 1900; |
| $mon++; |
| $mon = &normalizarValorDeUnDigito($mon); |
| $mday = &normalizarValorDeUnDigito($mday); |
| $hour = &normalizarValorDeUnDigito($hour); |
| $min = &normalizarValorDeUnDigito($min); |
| $sec = &normalizarValorDeUnDigito($sec); |
| my $descriptor = $year.$mon.$mday.$hour.$min.$sec; |
|  |
| # si me pidieron guardar el reporte creo el archivo |
| if ( $GUARDAR\_REPORTE ) |
| { |
| open(REPORTEX,">>$REPODIR/ReporteX\_$tipo\_rep.$descriptor") || die "ERROR: No puedo abrir el fichero ReporteX\_$tipo\_rep.$descriptor\n"; |
| print REPORTEX "$titulos"; |
| } |
|  |
| foreach $elem (@reporte) |
| { |
| chomp; |
| printf ("%-15s",$elem); |
|  |
| # guardo si me lo pidieron |
| if ( $GUARDAR\_REPORTE ) |
| { |
| printf ( REPORTEX "%-15s",$elem); |
| } |
|  |
| $cantCol++; |
|  |
| # paso al siguiente renglon |
| if ( $cantCol == 4 ) |
| { |
| print "\n"; |
| if ( $GUARDAR\_REPORTE ) |
| { |
| print REPORTEX "\n"; |
| } |
| $cantCol = 0; |
| $cantReg++; |
| } |
| } |
|  |
| # imprimo la cantidad de registros |
| print "\n"; |
| printf("%d registros",$cantReg); |
| print "\n"; |
|  |
| if ( $GUARDAR\_REPORTE ) |
| { |
| print REPORTEX "\n"; |
| printf (REPORTEX "%d registros",$cantReg); |
| print REPORTEX "\n"; |
| close(REPORTEX); |
| } |
| } |
|  |
| # |
| # Muestra el reporte correspondiente según el parametro enviado |
| # |
| sub mostrarReportes{ |
|  |
| my @valores\_registro; |
|  |
| my $pais; |
| my $sistema; |
| my $anio; |
| my $parametro; |
| my $comando; |
|  |
| my $subs; |
| my $rta; |
| my $parametro\_de\_rangos=0; |
| my $parametro\_de\_pais=0; |
| my $parametro\_val\_ref=0; |
| my $parametro\_sistema=0; |
|  |
| # inicializo las variables globales |
| &inicializarGlobales; |
|  |
| # tomo los parametros y los convierto a strings |
| my $aux = join(' ',@\_); |
|  |
| # quito caracter de eol |
| chomp($aux); |
|  |
| # paso parametros a array |
| my @parametros = split(/ /,$aux); |
|  |
| # obtengo el comando de la subrutina |
| $comando = @parametros[0]; |
|  |
| # elimino el comando correspondiente a la subrutina |
| splice(@parametros, 0, 1); |
|  |
| # me fijo los parametros enviados |
| foreach $parametro( @parametros ) |
| { |
| # el primer parametro es el pais, que no lleva identificador |
| # porque va siempre...si no encuentra un igual antes, es ese... |
| if ( $parametro =~ m/-p=/ ) |
| { |
| if ( $parametro\_de\_pais == 0 ) |
| { |
| $PAIS\_DESC = substr $parametro, index($parametro,'=')+1, (length $parametro)-3; |
| $parametro\_de\_pais = 1; |
| $COMANDOS\_USADOS.="País: $PAIS\_DESC. "; |
| } |
| else |
| { |
| print "Puede indicar solo un país por consulta.\n"; |
| return(0); |
| } |
| } |
| elsif ( $parametro =~ m/x=/ ) |
| { |
| if( uc($comando) eq uc("-dp") or uc($comando) eq uc("-dm") ) |
| { |
| if ( $parametro\_val\_ref == 0 ) |
| { |
| $VALOR\_DIFERENCIA = sprintf("%.2f",(substr $parametro, index($parametro,'=')+1, (length $parametro)-2)); |
| $COMANDOS\_USADOS.="Valor Diferencia: $VALOR\_DIFERENCIA. "; |
| $parametro\_val\_ref = 1; |
| } |
| else |
| { |
| print "Puede indicar solo un valor para diferencias por consulta.\n"; |
| return(0); |
| } |
| } |
| else |
| { |
| print "El comando ingresado no admite el parametro \"x\".\n"; |
| return(0); |
| } |
| } |
| elsif ( $parametro =~ m/-s=/) |
| { |
| if ( $parametro\_sistema != 0 ) |
| { |
| print "Puede indicar solo un sistema por consulta.\n"; |
| return(0); |
| } |
| else |
| { |
| $SIS\_ID = substr $parametro, index($parametro,'=')+1, (length $parametro)-3; |
| $parametro\_sistema=1; |
| $COMANDOS\_USADOS.="Id del Sistema: $SIS\_ID. "; |
| } |
| } |
| elsif ( $parametro =~ m/-a=/) |
| { |
| if ( $parametro\_de\_rangos != 0 ) |
| { |
| print "Puede indicar solo un año, periodo o rango de periodos por consulta.\n"; |
| return(0); |
| } |
| else |
| { |
| $CTB\_ANIO = substr $parametro, index($parametro,'=')+1, (length $parametro)-3; |
| $parametro\_de\_rangos = 1; |
| $COMANDOS\_USADOS.="Año: $CTB\_ANIO. "; |
| } |
| } |
| elsif ( $parametro =~ m/-pe=/ ) |
| { |
| if ( $parametro\_de\_rangos != 0 ) |
| { |
| print "Puede indicar solo un año, periodo o rango de periodos por consulta.\n"; |
| return(0); |
| } |
| else |
| { |
| $PERIODO = substr $parametro, index($parametro,'=')+1, (length $parametro)-4; |
| $parametro\_de\_rangos = 1; |
| $COMANDOS\_USADOS.="Período: $PERIODO. "; |
| } |
| } |
| elsif ( $parametro =~ m/-rp=/ ) |
| { |
| if ( $parametro\_de\_rangos != 0 ) |
| { |
| print "Puede indicar solo un año, periodo o rango de periodos por consulta.\n"; |
| return(0); |
| } |
| else |
| { |
| $RANGO\_PERIODOS = substr $parametro, index($parametro,'=')+1, (length $parametro)-4; |
| $parametro\_de\_rangos = 1; |
| $COMANDOS\_USADOS.="Rango de Períodos: $RANGO\_PERIODOS. "; |
| } |
| } |
| elsif ( $parametro =~ m/-g/) |
| { |
| $GUARDAR\_REPORTE = 1; |
| $COMANDOS\_USADOS.="El reporte fue guardado. "; |
| } |
| else |
| { |
| print "El parametro $parametro es incorrecto.\n"; |
| return(0); |
| } |
| } |
|  |
| if ( ! $parametro\_de\_pais ) |
| { |
| print "Debe indicar un país.\n"; |
| return(0); |
| } |
|  |
| # busco el codigo del pais ingresado |
| &buscarDatosPais; |
|  |
| # si no se encontro el país, salgo de la subrutina |
| if ( $PAIS\_ID eq "" ) |
| { |
| print "El país ingresado no está en la base de datos."; |
| return(0); |
| } |
|  |
| # obtengo los prestamos impagos segun los filtros |
| &obtenerPrestamosImpagos; |
|  |
| # obtengo los prestamos de cada pais |
| &obtenerPrestamosPais; |
|  |
| # muestro el reporte correspondiente |
| if ( $comando eq "-cr" ) |
| { |
| # muestro los resultados y recomendaciones |
| &mostrarRecomendacion; |
|  |
| # grabo si corresponde |
| if (&confirma("\n¿Desea grabar el recalculo? [s/n]: ")) |
| { |
| &grabarRecalculo("RECALCULO.".$PAIS\_DESC,7); |
| } |
| } |
| elsif ( $comando eq "-dp" || $comando eq "-dm" ) |
| { |
| # muestro las diferencias |
| &mostrarDiferencia($comando); |
| } |
| } |
|  |
| # analizo los valores de los parametros que me pasaron |
| sub analizarParametros{ |
|  |
| #inicializo las variables |
| my $rta = "s"; |
| my $cmd; |
| my $pmt; |
| my $respuesta; |
|  |
| # obtengo los valores pasados como parametros |
| $cmd = @\_[0]; |
|  |
| if ($cmd eq "-a") |
| { |
| &mostrarAyuda(@\_[1]); |
| } |
| elsif ($cmd eq "-cr" || $cmd eq "-dp" || $cmd eq "-dm" ) |
| { |
| &mostrarReportes(@\_); |
| } |
| else |
| { |
| print "El comando es incorrecto.\n"; |
| } |
|  |
| return &confirma("\n¿Desea realizar otra consulta? [s/n]: "); |
|  |
| } |
|  |
| ###################### |
| #                    # |
| # PROGRAMA PRINCIPAL # |
| #                    # |
| ###################### |
|  |
| # primero obtengo los directorios necesarios para el procesamiento |
| &obtenerDirectorios; |
| # tomo los parametros y los convierto a strings |
| my $aux = join(' ',@ARGV); |
| # quito caracter de eol |
| chomp($aux); |
| # paso parametros a array |
| my @parametros = split(/ /,$aux); |
|  |
| # muestra resultado de consultas, hasta que |
| # el usuario indique que quiere terminar |
| my $seguir = &analizarParametros(@parametros); |
| while ( $seguir ) |
| { |
| print "./ReporteX.pl "; |
| $aux = <STDIN>; |
| chomp($aux); |
| @parametros = split(/ /,$aux); |
| $seguir = &analizarParametros(@parametros); |
| } |

4.6 - GlogX.sh

**Título:** GlogX

**Justificación**

Centralizar el registro de mensajes en logs, respetando el formato W5 y controlando el espacio utilizado por dicho log.

**Archivos de Input, Intermedios y de Output**

No posee.

**Parámetros y Opciones**

Recibe 4 parámetros:

- El comando que desea grabar un mensaje en un log.

- El Tipo de mensaje: I 🡪 INFORMATIVO, W 🡪 WARNING, E 🡪 ERROR, SE 🡪 SEVERAL ERROR.

- El mensaje a grabar.

- El nombre del archivo de log.

**Invocación Manual y/o automática con ejemplos**

Ejemplo1:

./GlogX "InterpreteX" "SE" "No se encontró el archivo ps-mae" "InterpreteX"

Registrará el mensaje: "No se encotró el archivo ps-mae" en el log InterpreteX.log (asumiendo que es la extensión definida por el usuario) con el siguiente formato:

Fecha Usuario InterpreteX SE "No se encontró el archivo ps-mae"

**Hipótesis y Aclaraciones especificas del comando**

- El cuarto parámetro tiene como intención facilitar al usuario el uso de otro nombre, distinto del del comando, para el log. Por ejemplo, podría querer almacenar los mensajes informativos y los warnings en un log y en otro los errores normales y severos.

- Para evitar repetir código, se verifica si el comando invocante es InstalarX, en cuyo caso se setean las variables correspondientes para dicho comando.

**Código fuente**

#! /bin/bash

# 4to parametro: nombre del archivo de log sin extension

function GlogX {

# Cantidad de lineas a dejar

nroLineas=50

# Si no se pasan los 4 parametros solicitados, es un error

if [ $# -eq 5 ]

then

return 1

fi

# Chequeo si están seteadas las variables de ambiente

if [ -z $LOGEXT ] && [ -z $LOGDIR ] && [ -z $LOGSIZE ] && [ -z $CONFDIR ]

then

return 1

fi

comando=$1

tipoMensaje=$2

mensaje=$3

archivoLog=$4

directorioLog=$LOGDIR

fecha=$(date)

usuario=$(whoami)

if [ "$comando" = "InstalarX.sh" ]

then

directorioLog=$CONFDIR

LOGEXT=".log"

else

# Si no es el log de instalacion, veo el tamaño

if [ -f $directorioLog/$archivoLog$LOGEXT ] && [ `du -k $directorioLog/$archivoLog$LOGEXT | awk '{ print $1 }'` -gt $LOGSIZE ]

then

tail -50 $directorioLog/$archivoLog$LOGEXT > $directorioLog/$archivoLog$LOGEXT

echo $fecha $usuario $comando $tipoMensaje "Log Excedido" >> $directorioLog/$archivoLog$LOGEXT

fi

fi

# Escribo el log

# When Who Where What Why

echo $fecha $usuario $comando $tipoMensaje $mensaje >> $directorioLog/$archivoLog$LOGEXT

return 0

}

**4.7 - VlogX.sh**

**Título:** VlogX

**Justificación**

Permitir visualizar los logs en forma simple y ordenada, sin tener que buscar en todo el archivo.

**Archivos de Input, Intermedios y de Output**

No posee.

**Parámetros y Opciones**

Recibe 3 parámetros

- El comando invocante.

- El filtro que puede ser:

- Un número que indica la cantidad de líneas que se desean ver.

- Un string que se desea buscar en el log.

- El nombre del archivo de log.

**Invocación Manual y/o automática con ejemplos**

Ejemplo 1:

./VlogX.sh "Console" -10 "InicioX"

Mostrará las últimas 10 líneas del log de InicioX.

Ejemplo 2:

./VlogX.sh "Console" "no se encontró" "InicioX"

Mostrará las líneas del log de InicioX que contengan la frase "no se encontró".

**Hipótesis y Aclaraciones especificas del comando**

No aplica.

**Código fuente**

#! /bin/bash

# Filtros:

# -n: muestra las ultimas n lineas

# string: filtra las lineas que tienen el string

function VlogX {

# Si no se pasan los 4 parametros solicitados, es un error

if [ $# -eq 4 ]

then

return 1

fi

comando=$1

filtro=$2

archivoLog=$3

directorioLog=$LOGDIR

usuario=$(whoami)

if [ comando = "InstalarX.sh" ]

then

directorioLog=$CONFDIR

fi

# Leo el log segun el filtro

if [[ $filtro =~ \-[0-9]+ ]]

then

tail $filtro $directorioLog/$archivoLog$LOGEXT

else

cat $directorioLog/$archivoLog$LOGEXT | grep -i $filtro

fi

return 0

}

VlogX $1 $2 $3

**4.8 - StartX.sh**

**Título:** StartX.sh

**Justificación:**

Centralizar el inicio de procesos desde una función general, la cual verifica si no está actualmente en ejecución y en caso que no lo esté, lo lanza en background.

**Archivos de Input, Intermedios y de Output**

No posee.

**Parámetros y Opciones**

Recibe 2 parámetros, el primero, indica el comando que lo invocó y su única finalidad es que pueda ser registrado en el log. El segundo, es el comando cuya invocación se debe realizar.

**Invocación Manual y/o automática con ejemplos:**

Ejemplo 1:

./StartX.sh "Console" "./InstalarX.sh"

Ejecutará el script InstalarX.sh y dejará grabado en el log que el comando invocante es "Console"

Ejemplo 2:

./StartX.sh "InicioX" "./DetectaX.sh"

Ejecutará el script DetectaX.sh y dejará grabado en el log que el comando invocante es "InicioX"

**Hipótesis y Aclaraciones especificas del comando**

- Los comandos se ejecutan siempre en background, a menos que el comando a ejecutar sea DetectaX.

**Código fuente:**

#! /bin/bash

# Parametro 1: comando invocante

# Parametro 2: comando a ejecutar

function StartX {

source GlogX.sh;

if [ $# -eq 3 ]

then

return 1

fi

comando=$1

comandoAEjecutar=$2

nombreComando=$( echo "$comandoAEjecutar" | cut -f1 -d" ")

var=`ps -fea | grep -v "grep" | grep -v "StartX" | grep "$nombreComando" | wc -l`

if [ "$var" -ne 0 ]

then

GlogX "$comando.sh" "SE" "StartX.sh encuentra un proceso $nombreComando en ejecución" "$comando"

return 1

fi

GlogX "$comando.sh" "I" "StartX.sh invoca a $nombreComando" "$comando"

if [ $nombreComando == "DetectaX.sh" ]; then

$comandoAEjecutar &

else

$comandoAEjecutar

fi

return $!

}

StartX "$1" "$2"

**4.9 - StopX.sh**

**Título:** StopX

**Justificación**

Centralizar la finalización de procesos desde una función general, la cual verifica si no está actualmente en ejecución y en caso que no lo esté, lo lanza en background.

**Archivos de Input, Intermedios y de Output**

No posee.

**Parámetros y Opciones**

Recibe un único parámetro: el id del proceso a finalizar.

**Invocación Manual y/o automática con ejemplos**

./StopX 2368

Finaliza el proceso de id 2368, si es que existe.

**Hipótesis y Aclaraciones especificas del comando**

No posee

**Código fuente**

#! /bin/bash

# Parametro: id del proceso a finalizar

function StopX {

if [ $# -eq 2 ]

then

return 1

fi

proceso=$1

# Chequeo que el proceso exista

if [ `ps | awk '{ print $1 }' | grep $proceso | wc -l` -eq 0 ]

then

return 1

fi

kill $proceso

return 0

}

StopX $1

**4.10 - MoverX.sh**

**Título:** MoverX

**Justificación**

Centralizar el movimiento de archivos de un directorio a otro, verificando que no exista dicho archivo en el directorio destino y guardando los duplicados con un número de secuencia que indique el número de veces que se intentó copiar dicho archivo.

**Archivos de Input, Intermedios y de Output**

No posee.

**Parámetros y Opciones**

Recibe 3 parámetros:

- El archivo que se desea mover.

- El directorio al que se lo desea mover.

- El comando invocante.

**Invocación Manual y/o automática con ejemplos**

Ejemplo1:

./MoverX /home/archivo1.txt /bin "Console"

Mueve el archivo "archivo1.txt" del directorio /home al directorio /bin.

Ejemplo2:

./MoverX /home/archivo1.txt /bin "Console"

Mueve el archivo "archivo1.txt" del directorio /home al directorio /bin. Como dicho archivo ya existe, creará una carpeta llamada dup, donde almacenará "archivo1.txt" agregandole ".1" al final, para indicar que es el duplicado N° 1.

**Hipótesis y Aclaraciones especificas del comando**

- El primer parámetro contiene el directorio inicial y el nombre del archivo.

- El segundo parámetro sólo contiene el directorio destino.

**Código fuente**

#! /bin/bash

function MoverX () {

# Si no se pasan los 3 parametros solicitados, es un error

if [ $# -eq 4 ]

then

return 1

fi

# Tomo los parametros

origen=$1

destino=$2

comando=$3

carpetaOrigen=${origen%/\*}

archivo=${origen##\*/}

# 1.1 Verifico si la carpeta de origen y de destino son iguales

if [ $carpetaOrigen = $destino ]

then

return 1

fi

# 1.2 Verifico que existan el archivo origen y la carpeta destino

if [ ! -f $origen ] || [ ! -d $destino ]

then

return 1

fi

# 1.3 Verifico si es un archivo duplicado

if [ -f "$destino/$archivo" ]

then

# Verifico si existe la carpeta dup, si no, la creo y genero el numero de secuencia

if [ ! -d "$destino/dup" ] || [ $(ls -1 $destino/dup | wc -l) = 0 ]

then

mkdir -p "$destino/dup"

nroSecuencia="0"

else

maxNroArchivo=`ls -l $destino/dup/$archivo\* | tail -1 | awk '{ print $NF }' | awk -F. '{ print $NF }'`

let "nroSecuencia = maxNroArchivo + 1"

fi

# Copio el archivo a la carpeta de duplicados

mv $origen "$destino/dup/$archivo.$nroSecuencia"

return 1

fi

mv $origen $destino

return 0

}

5 – Archivos

Archivos del Enunciado

* Al adjuntar el enunciado en este informe asumimos que las estructuras de estos archivos quedan clara y correctamente especificadas en el mismo.

Archivos Auxiliares

* No aplica. No se han utlizados archivos auxiliares.

6 – Directorios, Datos y Salida

**Directorios**

El sistema tendrá en una primera instancia la siguiente estructura de directorios:

grupo01

README

instalacion

mae

p-s.mae

PPI.mae

tablas

T1.tab

T2.tab

bin

InicioX.sh

DetectaX.sh

valPais.sh

Interprete.sh

ReporteX.sh

GlogX.sh

VlogX.sh

MoverX.sh

StartX.sh

StopX.sh

InstalarX.sh

Donde:

instalacion: contiene todos los archivos necesarios para ejecutar una correcta instalación del sistema.

instalacion/mae: contiene a los archivos maestros

instalacion/bin: contiene a los archivos ejecutables

instalación/tablas: contiene a las tablas de configuracion

Una vez realizada la instalación. Se crearán una serie de directorios para poder ejecutar CONTROLX. A continuación se muestra como será la estructura de directorios dentro de grupo01 realizando una instalación por default:

grupo01

aceptados

arribos

bin

InicioX.sh

DetectaX.sh

valPais.sh

Interprete.sh

ReporteX.sh

GlogX.sh

VlogX.sh

MoverX.sh

StartX.sh

StopX.sh

conf

T1.tab

T2.tab

InstalarX.conf

InstalarX.log

instalación

mae

tablas

bin

InstalarX.sh

log

mae

p-s.mae

PPI.mae

procesados

rechazados

reportes

7 – Apéndice

Enunciado

Índice

Enunciado – Tema X 80

Pasos 80

Instalación 80

Inicialización 80

Detección y Recepción 80

Intérprete 80

Consultas y Reportes 80

Requerimiento 80

A. Desarrollar el Sistema CONTROLX compuesto por: 80

B. Elaborar la Documentación del Sistema CONTROLX compuesta por: 81

C. Entregar el Sistema y la documentación del sistema CONTROLX 82

Recomendaciones para el equipo de desarrollo 83

Recomendaciones para el equipo de integración y testing 83

Archivos y Estructuras 85

Archivo de Configuración 85

Tabla de Campos 85

Tabla de Separadores 85

Archivo Maestro Contable de Préstamos Personales Impagos 86

Archivo Maestro de Países y Sistemas 86

Archivos de Préstamos Personales (Novedades) 87

Archivos de Préstamos Personales por País 88

Archivos de Recálculo 89

Archivos de Reporte 89

Archivo de Log 89

Especificación de Comandos 90

Instalación: InstalX 90

Inicialización: InicioX 97

Detección y Recepción: DetectaX 99

Interprete 101

Consultas y Reportes: ReporteX 104

Especificación de Funciones 108

StartX 108

StopX 108

MoverX 109

GlogX 110

VlogX 111

# Enunciado – Tema

El área de control de inversiones de una multinacional desea implementar un sistema software denominado CONTROL que le permita ir procesando archivos que contienen información de préstamos personales impagos generada por diferentes sistemas de sus subsidiarias que operan en distintos países. Estos archivos se reciben, validan, interpretan, formatean, y se los deja listos para poder aplicarle diversas consultas.

## Pasos

### Instalación

El sistema software desarrollado debe tener un comando para su instalación. Esta actividad debe estar reflejada en el comando

### Inicialización

El Proceso se inicia con el aseguramiento de la disponibilidad de la información para llevar adelante el proceso total: Es indispensable contar con los comandos, archivos maestros y las tablas de configuración con los permisos adecuados.

Continua con la configuración de las variables de ambiente y otras variables especiales para la ejecución del siguiente comando. Ofrece arrancar automáticamente el comando de recepción de archivos. Todas estas actividades deben estar reflejadas en el comando .

### Detección y Recepción

Luego tenemos el proceso de recepción de los archivos de préstamos. Cada país/sistema envía los archivos por año/mes (periodo). Se deberá detectar el arribo de los archivos a un directorio específico que el usuario define en el momento de la instalación y validar el nombre de estos archivos. Arranca, si corresponde, al Intérprete. Esta actividad debe estar reflejada en el comando .

### Intérprete

Los archivos aceptados no poseen una estructura estándar. La misma se encuentra definida a través de las tablas de separadores y campos que tiene el sistema. Para realizar la lectura de los archivos se debe interpretar su información a través de estas tablas y generar una estructura estándar que pueda ser usada por el siguiente comando. Esta actividad debe estar reflejada en el comando Interprete.

### Consultas y Reportes

Por último tenemos el proceso de consulta y emisión de listados según las opciones y parámetros ingresados. Esta actividad debe estar reflejada en el comando perl .

## Requerimiento

### Desarrollar el Sistema CONTROL compuesto por:

* Shell script denominado que efectúa la inicialización de ambiente.
* Shell script denominado que detecta el arribo de archivos de préstamos.
* Shell script denominado que genera los archivos formateados.
* Programa Perl denominado que resuelve las consultas y emite informes.
* Función (en Shell o en Perl) denominada que se emplea para mover archivos
* Función (en Shell o en Perl) denominada que se emplea para grabar los archivos de log
* Función (en Shell o en Perl) denominada para mostrar por pantalla los registros del log en forma amigable.
* Función en Shell script denominada .que se emplea para detener procesos y otra complementaria que permite disparar procesos
* Shell script denominado que permite la instalación de todo lo anterior

### Elaborar la Documentación del Sistema CONTROL compuesta por:

1. **Carátula**

La entregada en este mismo documento con los datos completos en 2 COPIAS una para el grupo y otra para el docente.

1. **Planillas de Evaluación**

Las entregadas en este mismo documento

1. **Índice del Contenido de la Carpeta.**

El número de página puede ser incorporado manualmente luego de numerar las hojas

1. **Hipótesis y Aclaraciones Globales**

Describa las hipótesis que ha considerado para la resolución del TP. Documente cualquier otra aclaración que se considere necesaria. Si en lugar de presentar las Hipótesis en un único capitulo, lo hace en cada comando, aclararlo en este punto.

1. **Problemas relevantes**

Describa los problemas relevantes que se hayan presentado durante el desarrollo, la integración y/o la prueba del sistema. Explique cómo fueron solucionados

1. **Archivo README**

Incluya la impresión del README en la carpeta. Ver detalles en “Recomendaciones para el equipo de integración y testing” y en la especificación del instalador.

1. **Listado de Comandos y Funciones**

Incluya todos los comandos y funciones desarrollados, para cada uno de ellos indique:

1. Título: Nombre del comando
2. Justificación, brinde una justificación de su creación, explique para que lo usa y dónde, solo para aquellos comandos que no fueron solicitados.
3. Archivos de Input, Intermedios y de Output
4. Parámetros y Opciones
5. Invocación Manual y/o automática con ejemplos
6. Hipótesis y Aclaraciones especificas del comando
7. Código fuente, el cual debe deberá incluir comentarios útiles para la comprensión del código (ver detalles en “Recomendaciones para el equipo de desarrollo”)
8. **Archivos**
9. Archivos del enunciado: Por cada archivo del enunciado indicar:
   * 1. Nombre del Archivo y Estructura.
10. Archivos auxiliares: Por cada archivo auxiliar que se utilice indicar:
    * 1. Nombre del Archivo y Estructura
      2. Justificación de su uso
11. **Directorios, Datos y Salida**
12. Directorios. Brinde un esquema de los directorios que usa el sistema
13. Incluya los datos de los archivos maestros
14. Muestre las Salidas que se van obteniendo con el caso de prueba entregado
15. Indique paso a paso:

* Que comando está ejecutando
* Con que datos (a nivel de archivo o a nivel de registro cuando es necesario) está procesando
* Cuál es el resultado obtenido (a nivel de archivo o a nivel de registro cuando es necesario)

1. Para el caso del comando de Impresión, muestre distintitas invocaciones y salidas que se obtienen indicando que es lo que se toma como input
2. **Apéndice**
3. Incluya el ENUNCIADO completo del TP (sin la caratula ni las planillas de evaluación dado que ya fueron incluidas al principio de la carpeta)

### Entregar el Sistema y la documentación del sistema CONTROL

El día de vencimiento del TP, cada ayudante convocará a los integrantes de un grupo e iniciará la corrección mediante una entrevista grupal. Es imprescindible la presencia de todos los integrantes del grupo el día de la corrección. El objetivo de esto es comprender la dinámica de trabajo del equipo y los roles que ha desempeñado cada integrante del equipo. Se solicitará la entrega de:

El sistema desarrollado

Deberá estar contenido en un único archivo instalable en formato “.tgz” con todos los archivos y directorios empaquetados en un archivo “tar” y luego comprimido con “gzip”. El instalable deberá contener:

* El archivo README
* Los scripts desarrollados
* Los Archivos de prueba, maestros y tablas de configuración
* Los Archivos de salida de las Pruebas ejecutadas
* Otros casos de prueba
* Cualquier otro archivo que se considere necesario

MUY IMPORTANTE: cuando se efectúa el “tar” se debe usar la opción de “directorio relativo” para evitar problemas cuando se proceda a la instalación en los laboratorios de la facultad.

La carpeta con la documentación del Sistema

La documentación debe entregarse en una carpeta con TODAS las hojas numeradas y enganchadas. Las hojas sueltas no se considerarán como parte de la misma

El pie de página de cada hoja debe tener: Número de Grupo y Tema (en el margen izquierdo) y Número de Hoja (en el margen derecho). La numeración puede ser manual

Evaluación

Se evaluará el trabajo grupal y a cada integrante en forma individual.

Para que el alumno apruebe el trabajo práctico debe estar aprobado en los dos aspectos.

Posterior a la entrega del TP se podrá acordar entre el Ayudante y el Grupo el intercambio de correspondencia a través de la cuenta [so7508@gmail.com](mailto:so7508@hotmail.com). Por este medio el ayudante podrá solicitar a los alumnos correcciones, mejoras, nuevos casos de prueba, etc. Cabe aclarar que todos los mensajes a esta casilla deberán tener como asunto “Grupoxx (xx es el número asignado al grupo) para poder redireccionar el mail al ayudante correspondiente.

Dentro de los ítems a chequear el ayudante evaluará aspectos formales (como ser la forma de presentación de la carpeta), aspectos funcionales: que se resuelva el problema planteado y aspectos operativos: que el TP funcione integrado.

### Recomendaciones para el equipo de desarrollo

1. Se deberá tener en cuenta para la resolución TODAS las condiciones que se enuncian.
2. Se deben respetar los formatos de archivos especificados
3. Se debe respetar la estructura de directorios planteada
4. Se debe evitar el uso de archivos auxiliares permanentes
5. Si se emplean archivos auxiliares temporales, se deben eliminar ANTES de finalizar la ejecución del comando
6. Código de Retorno

Toda invocación desde un comando a otro debe devolver un código de retorno cero (0) si fue exitoso o distinto de cero si tuvo errores. Siempre al finalizar el comando se debe loguear si finalizó correctamente o con errores.

1. Movimiento de Archivos

En líneas generales no se borra ningún archivo de datos, se los mueve de un lugar a otro para asegurar la integridad de la información original. Se solicita una función de librería para el movimiento de archivos de datos la cual debe ser empleada por todos los comandos que la requieran

1. Manejo de errores, logueo

Todo evento que genera algún tipo de error debe ser logueado. Si el comando no graba en un log especifico, muestre el error por pantalla. Evite detener el proceso de evaluación del TP por no poder detectar que está pasando. También puede programar al comando con una opción de ejecución que prende o apaga una señal que permite ir dejando pistas de auditoría en el log o por pantalla para conocer como se avanza en la ejecución del sistema.

1. Documentación del Código Fuente, responsabilidad del equipo de desarrollo

Dentro de los scripts se deben documentar las acciones que se están llevando a cabo a fin de facilitar la tarea de evaluación. Mantener una indentación homogénea en todos los scripts del mismo tipo para facilitar la lectura del código. Nombres de variables: mantener un estándar de nomenclatura de variables homogénea en todos los scripts

### Recomendaciones para el equipo de integración y testing

1. Se recomienda efectuar un diagrama global del proceso completo que muestre el mapa de servicios de cada componente y los responsables del desarrollo.
2. Prevean efectuar la prueba de integración varios días antes de la entrega del TP dado que usualmente surgen errores de comunicación entre los comandos encadenados. Recuerden que la entrega del TP INCLUYE que muestren el resultado obtenido luego de la ejecución con el caso de prueba provisto.
3. No es estrictamente necesario haber completado todo el código interno del comando para poder efectuar un ensayo de integración
4. El TP debe correr en las instalaciones de la FIUBA, por lo tanto una integración en sus equipos personales no garantiza que dentro del entorno de la FIUBA la integración muestre los mismos resultados. Ejemplo de esto es el retorno que da el comando ps.
5. Recuerde que dentro del entorno de ejecución de la FIUBA su usuario NUNCA puede ser ROOT
6. Lotes de Prueba

Se proveerán los archivos maestros, las tablas de configuración y algunos archivos de préstamos, todos con información libre de errores. No se proveerán registros con error. El grupo debe traer un juego extra de archivos de para realizar pruebas con casos lo suficientemente heterogéneos como para contemplar todas las variantes de ejecución, en particular las de rechazo o error

1. Archivo README, responsabilidad del equipo de integración

Como parte de la documentación del sistema se debe proveer un archivo README en donde se brinden indicaciones de instalación y de ejecución de los comandos, como ser:

1. Una explicación de cómo copiar desde un medio externo el instalable, a modo de ejemplo:

* Insertar el dispositivo de almacenamiento con el contenido del tp (pen drive, cd, etc)
* Crear en el directorio corriente un directorio de trabajo
* Copiar el archivo \*.tgz en ese directorio
* Descomprimir el \*.tgz de manera de generar un \*.tar
* Extraer los archivos del tar.
* … etc.

1. Instrucciones de instalación
2. Que se requiere para poder instalar, Que nos deja la instalación y donde
3. Cuáles son los primeros pasos para poder correr el paquete una vez instalado
4. Que comprobaciones se pueden hacer para asegurar que todo está en condiciones para empezar
5. Como frenar la ejecución de comandos

# Archivos y Estructuras

## Archivo de Configuración

#### Repositorio: CONFDIR/.conf

|  |  |  |
| --- | --- | --- |
| **Campos** | **Descripción** | **Fuente/Valor** |
| **Variable** | Caracteres | Valores posibles: GRUPO, CONFDIR,BINDIR,MAEDIR,ARRIDIR, ACEPDIR,RECHDIR, PROCDIR, REPODIR, LOGDIR, LOGEXT, LOGSIZE, DATASIZE |
| **Valor** | Caracteres | Contenido de la variable al momento de la grabación. |
| **Usuario** | Caracteres | Es el login del usuario que graba el registro |
| **Fecha** | Fecha y hora  Formato a Elección | Es la fecha y hora en el momento de grabación del registro. |

Separador de campos: = igual

Ejemplo: GRUPO=/usr/prueba01/temp/grupo01/tp=prueba01=30/03/2013 10:03 p.m

Luego de la grabación de estos registros, se pueden agregar registros con la estructura y contenido que el desarrollador desee. S**on de libre disponibilidad** para los desarrolladores.

En la función Mover, por ejemplo, deben utilizar secuenciadores. Si desean registrar estas secuencias, pueden usar este mismo archivo preservando los registros de configuración aquí detallados.

## Tabla de Campos

#### Repositorio: CONFDIR /T2.tab

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Variable** |
| Código de País | carácter | PAIS\_ID |
| Código de Sistema | numérico | SIS\_ID |
| Nombre del Campo | caracteres | NOM\_CAMP |
| Orden de aparición del campo | numérico | ORD\_CAMP |
| Formato del campo | caracteres | FMT\_CAMP |

Separador de campos: - guion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ejemplo DATOS T2.tab** | | | | |
| País | Sistema | Nombre del Campo | Nro. de Orden | Formato |
| A | 6 | MT\_IMPAGO | 1 | commax16.2 |
| A | 6 | MT\_PRES | 2 | commax16.2 |
| A | 6 | MT\_DEB | 3 | commax16.2 |
| A | 6 | MT\_INDE | 4 | commax16.2 |
| A | 6 | PRES\_CLI | 5 | $44. |
| A | 6 | CTB\_ESTADO | 6 | $4. |
| A | 6 | PRES\_ID | 7 | $12. |
| A | 6 | PRES\_CLI\_ID | 8 | $6. |
| A | 6 | MT\_INNODE | 9 | commax16.2 |
| A | 6 | CTB\_FE | 10 | ddmmyy10. |

## Tabla de Separadores

#### Repositorio: CONFDIR /T1.tab

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Variable** |
| Código de País | carácter | PAIS\_ID |
| Código de Sistema | numérico | SIS\_ID |
| Carácter Separador de Campo | carácter | SEP\_CAMP |
| Carácter Separador de decimales | carácter | SEP\_DEC |

Separador de campos: - guion

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejemplo DATOS T1.tab** | | | |
| **País** | **Sistema** | **Separador de Campo** | **Separador Decimal** |
| A | 6 | ; | , |

## Archivo Maestro Contable de Préstamos Personales Impagos

#### Repositorio: MAEDIR/PPI.mae

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Variable** |
| Código de País | carácter | PAIS\_ID |
| Código de Sistema | numérico | SIS\_ID |
| Año contable | numérico | CTB\_ANIO |
| Mes contable | numérico | CTB\_MES |
| Día contable | numérico | CTB\_DIA |
| Estado contable | caracteres | CTB\_ESTADO |
| Fecha Préstamo | Fecha | PRES\_FE |
| Código Préstamo | caracteres | PRES\_ID |
| Tipo Préstamo | caracteres | PRES\_TI |
| Monto Préstamo | numérico | MT\_PRES |
| Monto Impago | numérico | MT\_IMPAGO |
| Monto Interés Devengado | numérico | MT\_INDE |
| Monto Interés No Devengado | numérico | MT\_INNODE |
| Monto Debitado | numérico | MT\_DEB |

Separador de campos: ; punto y coma

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ejemplo DATOS PPI.mae** | | | | | | | | | | | | | |
| PAIS\_  ID | SIS\_  ID | CTB\_  ANIO | CTB\_  MES | CTB\_  DIA | CTB\_  ESTADO | PRES\_FE | PRES\_ID | PRES\_  TI | MT\_  PRES | MT\_  IMPAGO | MT\_  INDE | MT\_  INNODE | MT\_  DEB |
| A | 6 | 2012 | 10 | 1 | MOR1 | 12/12/2009 | RETA1366267 | ALOA | 11221,36 | 93,63 | 138,54 | 415,61 | 0 |
| A | 6 | 2012 | 10 | 1 | MOR1 | 11/11/2011 | RETA1366268 | ALOA | 11775,50 | 15,51 | 138,54 | 415,61 | 0 |
| A | 6 | 2012 | 10 | 1 | MOR2 | 10/10/2010 | RETA1373910 | ALOA | 10551,99 | 233,52 | 124,14 | 372,43 | 0 |
| A | 6 | 2012 | 10 | 1 | SMOR | 09/09/2009 | RETA1384934 | ACB | 0 | 6,06 | 0 | 0 | 0 |
| A | 6 | 2012 | 10 | 1 | SMOR | 08/08/2008 | RETA1384932 | ACB | 0 | 181,35 | 0 | 0 | 0 |
| A | 6 | 2012 | 11 | 1 | MOR1 | 11/11/2011 | RETA1369516 | ALOA | 9929,48 | 37,67 | 116,82 | 350,46 | 197,34 |
| A | 6 | 2012 | 12 | 1 | SMOR | 07/07/2007 | RETA1385029 | ALOA | 0 | 191,43 | 0 | 0,01 | 0 |
| A | 6 | 2012 | 12 | 1 | SMOR | 06/06/2006 | RETA1385022 | ACB | 0 | 0,07 | 0 | 0,01 | 0 |

## Archivo Maestro de Países y Sistemas

#### Repositorio: MAEDIR/p-s.mae

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Variable** |
| Código de País | carácter | PAIS\_ID |
| Nombre del País | caracteres | PAIS\_DESC |
| Código de Sistema | numérico | SIS\_ID |
| Nombre del Sistema | caracteres | SIS\_DESC |

Separador de campos: - guion

|  |  |  |  |
| --- | --- | --- | --- |
| **Ejemplo DATOS P-S.mae** | | | |
| País | Nombre del Pais | Sistema | Nombre del Sistema |
| A | Argentina | 6 | Oracle Financial |

## Archivos de Préstamos Personales (Novedades)

#### Directorio de arribos: ARRIDIR/nombre del archivo

Sin estructura definida

#### Repositorio de Archivos Rechazados: RECHDIR/nombre del archivo

Sin estructura definida

#### Repositorio de Archivos Aceptados: ACEPDIR/nombre del archivo

Estructura definida según T2.tab

#### Repositorio de Archivos Procesados: PROCDIR/nombre del archivo

Estructura definida según T2.tab

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ejemplo: novedades de Argentina, Oracle, 2012, Octubre** | | | | | | | | | |
| MT\_IMPAGO | MT\_PRES | MT\_  DEB | MT\_  INDE | PRES\_CLI | CTB\_  ESTADO | PRES\_ID | PRES\_  CLI\_ID | MT\_  INNODE | CTB\_FE |
| 93,63 | 11221,36 | 0 | 138,54 | ARGENTINA POINT CLAIM MANAGEMENT SRL | MOR1 | RETA1366267 | 783920 | 415,61 | 01/10/2012 |
| 15,51 | 11775,5 | 0 | 138,54 | ARGENTINA POINT CLAIM MANAGEMENT SRL | MOR1 | RETA1366268 | 783920 | 415,61 | 02/10/2012 |
| 233,52 | 10551,99 | 0 | 124,14 | ARGENTINA COACHWORKS | MOR2 | RETA1373910 | 722973 | 372,43 | 03/10/2012 |
| 6,06 |  |  |  | ARGENTINA CARS SA | MOR1 | RETA1384934 | 741508 |  | 04/10/2012 |
| 181,35 |  |  |  | ARGENTINA CARS SA | MOR1 | RETA1384932 | 741508 |  | 05/10/2012 |
| ***Ejemplo: novedades de Argentina, Oracle, 2012, Noviembre*** | | | | | | | | | |
| 37,67 | 9929,48 | 197,34 | 116,82 | ARGENTINA DYNE MOTORS SA | MOR1 | RETA1369516 | 722973 | 350,46 | 06/11/2012 |
| **Ejemplo: novedades de Argentina, Oracle, 2012, diciembre** | | | | | | | | | |
| 191,43 | 0,01 |  |  | AUTO ARGENTINA LANGUAGE SA | MOR1 | RETA1385029 | 541165 | 0,01 | 07/12/2012 |
| 0,07 | 0,01 |  |  | AUTO ARGENTINA LANGUAGE SA | MOR1 | RETA1385022 | 541165 | 0,01 | 08/12/2012 |

## Archivos de Préstamos Personales por País

#### Repositorio: PROCDIR/prestamos.<país>

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Tipo** | **Variable** | **Fuente** |
| Código de Sistema | numérico | SIS\_ID | filename |
| Año contable | numérico | CTB\_ANIO | Input CTB\_FE[anio] |
| Mes contable | numérico | CTB\_MES | Input CTB\_FE[mes] |
| Día contable | numérico | CTB\_DIA | Input CTB\_FE[dia] |
| Estado contable | caracteres | CTB\_ESTADO | Input.CTB\_ESTADO |
| Código Préstamo | caracteres | PRES\_ID | Input.PRES\_ID |
| Monto Préstamo | numérico | MT\_PRES | Input.MT\_PRES |
| Monto Impago | numérico | MT\_IMPAGO | Input.MT\_IMPAGO |
| Monto Intereses Devengados | numérico | MT\_INDE | Input.MT\_INDE |
| Monto Intereses No Devengados | numérico | MT\_INNODE | Input.MT\_INNODE |
| Monto Debitado | numérico | MT\_DEB | Input.MT\_DEB |
| Monto Restante | numérico | MT\_REST | Fórmula |
| Código del Cliente | caracteres | PRES\_CLI\_ID | Input.PRES\_CLI\_ID |
| Nombre del Cliente | caracteres | PRES\_CLI | Input.PRES\_CLI |
| Fecha de Grabación | Fecha | INS\_FE |  |
| Usuario de Grabación | caracteres | INS\_USER |  |

Separador de campos: ; punto y coma

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Ejemplo: algunos prestamos de argentina*** | | | | | | | | | | | | | | | |
| SIS\_ID | CTB\_  ANIO | CTB\_MES | CTB\_  DIA | CTB\_  ESTADO | PRES\_  ID | MT\_  PRES | MT\_  IMPAGO | MT\_  INDE | MT\_  INNODE | MT\_  DEB | MT\_  REST | PRES\_  CLI\_ID | PRES\_  CLI | INS\_  FE | INS\_  USER |
| 6 | 2012 | 10 | 1 | MOR1 | RETA1366267 | 11221,36 | 93,63 | 138,5 | 415,6 | 0 | 11869,14 | 783920 | ARGENTINA POINT CLAIM MANAGEMENT SRL | 09/04/2013 | Prueba01 |
| 6 | 2012 | 10 | 2 | MOR1 | RETA1366268 | 11775,5 | 15,51 | 138,5 | 415,6 | 0 | 12345,16 | 783920 | ARGENTINA POINT CLAIM MANAGEMENT SRL | 09/04/2013 | Prueba01 |
| 6 | 2012 | 10 | 3 | MOR2 | RETA1373910 | 10551,99 | 233,5 | 124,1 | 372,4 | 0 | 11282,08 | 722973 | ARGENTINA COACHWORKS | 09/04/2013 | Prueba01 |
| 6 | 2012 | 10 | 4 | MOR1 | RETA1384934 | 0 | 6,06 | 0 | 0 | 0 | 6,06 | 741508 | ARGENTINA CARS SA | 09/04/2013 | Prueba01 |
| 6 | 2012 | 10 | 5 | MOR1 | RETA1384932 | 0 | 181,4 | 0 | 0 | 0 | 181,35 | 741508 | ARGENTINA CARS SA | 09/04/2013 | Prueba01 |
| 6 | 2012 | 11 | 6 | MOR1 | RETA1369516 | 9929,48 | 37,67 | 116,8 | 350,5 | 197,3 | 10237,09 | 722973 | ARGENTINA DYNE MOTORS SA | 09/04/2013 | Prueba01 |
| 6 | 2012 | 12 | 7 | MOR1 | RETA1385029 | 0,01 | 191,4 | 0 | 0,01 | 0 | 191,45 | 541165 | AUTO ARGENTINA LANGUAGE SA | 09/04/2013 | Prueba01 |
| 6 | 2012 | 12 | 8 | MOR1 | RETA1385022 | 0,01 | 0,07 | 0 | 0,01 | 0 | 0,09 | 541165 | AUTO ARGENTINA LANGUAGE SA | 09/04/2013 | Prueba01 |

## Archivos de Recálculo

#### Repositorio: REPODIR/recalculo.<país>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Campo** | **Tipo** | **Variable** | | **Fuente** |
| Código de Sistema | numérico | SIS\_ID | | Archivo Maestro |
| Año contable | numérico | CTB\_ANIO | | Archivo Maestro |
| Mes contable | numérico | CTB\_MES | | Archivo Maestro |
| Día contable | numérico | CTB\_DIA | Archivo Prestamos | |
| Estado contable | caracteres | CTB\_ESTADO | Archivo Maestro | |
| Código Préstamo | caracteres | PRES\_ID | | Archivo Maestro |
| Monto Préstamo | numérico | MT\_PRES | | Archivo Maestro |
| Monto Impago | numérico | MT\_IMPAGO | | Archivo Maestro |
| Monto Interés Devengado | numérico | MT\_INDE | | Archivo Maestro |
| Monto Interés No Devengado | numérico | MT\_INNODE | | Archivo Maestro |
| Monto Debitado | numérico | MT\_DEB | | Archivo Maestro |
| Monto Restante | numérico | MT\_REST | El calculado del maestro | |
| Código del Cliente | caracteres | PRES\_CLI\_ID | | Archivo Prestamos |
| Nombre del Cliente | caracteres | PRES\_CLI | | Archivo Prestamos |
| Fecha de Grabación | Fecha | INS\_FE | |  |
| Usuario de Grabación | caracteres | INS\_USER | |  |

Separador de campos: ; punto y coma

## Archivos de Reporte

#### Repositorio: REPODIR/.xxx

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Fuente** |
| **Línea de Reporte** | Caracteres | Proceso . |

Separador de campos: no posee

## Archivo de Log

#### Repositorio: LOGDIR/<comando>.logext

|  |  |  |
| --- | --- | --- |
| **Campo** | **Descripción** | **Fuente/Valor** |
| **Cuando** | Fecha y hora  Formato a Elección | Es la fecha y hora en el momento de grabación del registro. |
| **Quien** | Caracteres | Es el login del usuario que graba el registro |
| **Donde** | Caracteres | Nombre del Comando, función o rutina en donde se produce el evento que se registra en el log |
| **Que** | Caracteres | Lo determina el programador. Valores Posibles: I,W,E,SE |
| **Porque** | Hasta 120 Caracteres | Lo determina el programador. |

Separador de campos: - guion

**Ejemplo:** 20130509 10:53:22-PRUEBA01--W-No se pudo arrancar el demonio

# Especificación de Comandos

A continuación se brinda una descripción de cada comando solicitado.

En esta descripción se sugieren los pasos de ejecución solo a los efectos de ordenar la explicación.

Estos pasos deben considerarse indicativos. Si el equipo de desarrollo lo considera pertinente, puede modificarlos tanto sea en el orden de ejecución como en la forma de resolverlo, siempre y cuando esto no afecte el resultado final esperado.

## Instalación:

Opciones y Parámetros

* A especificar por el desarrollador
* Explicar claramente su invocación y uso en el README.

Descripción

El propósito de este comando es efectuar la instalación del sistema CONTROL.

El script debe permitir instalar todos los comandos al mismo tiempo, o de a uno por vez.

Cuando este script interactúa con el usuario, los valores por default propuestos se deberán mostrar entre paréntesis (). En el arranque los valores default se indican en el enunciado, pero si el hilo de ejecución se reinicia en otro punto los valores default deben ser reemplazados por lo ingresado anteriormente en ese mismo paso por el usuario.

Los usuarios disponibles en el entorno de evaluación del TP (Laboratorios del 4to Piso) son solo tres: prueba01 (pw test01), prueba02 (pw test02), prueba03 (pw test03). Debido a esto cada grupo deberá emplear un directorio de trabajo llamado grupoxx (donde xx es el nro de grupo).

Antes de comenzar la instalación asegúrese que:

* la última versión del TP está grabada en el dispositivo externo
* que la terminal que se usara para la evaluación posea interfaz para la conexión del dispositivo

Una vez que conecte el dispositivo externo, deberá copiar su contenido en un subdirectorio de trabajo, descomprimir para generar un tar, extraer los archivos del tar, etc.

Ud. puede definir el espacio de trabajo como lo desee, excepto por el directorio de configuración.

La ubicación del directorio de configuración no puede ser variable, debe ser conocida para permitirle al comando de inicialización de ambiente encontrar el archivo de configuración, por lo cual se pide que la extracción genere AL MENOS el siguiente directorio:

*….…/grupoxx*/conf

Estos pasos y resultados deberán ser explicados claramente en el README.

Para simplificar esta narrativa de aquí en mas a todo el camino (path) que va desde la raíz hasta *grupoxx* lo llamaremos genéricamente **“$grupo”,** a $grupo/conf lo llamaremos CONFDIR, **etc**

Pasos sugeridos

###### Inicializar archivo de log

Este comando graba un archivo de Log cuyo nombre es .log en el directorio **CONFDIR**.

Si el archivo de log no existe, crearlo. Si existe, agregarle los nuevos registros.

Un log de instalación debe permitir recrear en forma completa el proceso de instalación, es decir: las condiciones antes de la instalación, las decisiones tomadas durante la instalación, las condiciones finales.

La escritura en el log de instalación debe respetar las condiciones indicadas en la función y debe poder ser visualizado con .

El primer mensaje que se debe grabar es el mensaje de inicio de ejecución (todos los mensajes que se muestran son a modo de ejemplo)

Inicio de Ejecución

###### Mostrar (y grabar en el log) donde se graba el log de la instalación

Log del Comando : CONFDIR/.log

###### Mostrar (y grabar en el log) el nombre del directorio de configuración

Directorio de Configuración: CONFDIR

###### Detectar si el paquete CONTROL o alguno de sus componentes ya está instalado

Este script de instalación se debe preparar de manera tal que detecte si el paquete o alguno de sus componentes ya se encuentran instalados

Una forma de hacer esto es verificando la existencia del archivo .conf

* Si .conf no existe, asumir que el paquete no fue instalado. El proceso sigue en el siguiente punto
* Si .conf existe, asumir que el paquete ya fue instalado y verificar que la instalación este completa, o indicar los faltantes
  1. Si esta completo, mostrar y grabar en el log los siguientes mensajes y luego ir a **FIN**

TP SO7508 . Tema Copyright © Grupo xx

Librería del Sistema: CONFDIR (mostrar path y listar archivos)

Ejecutables: BINDIR (mostrar path y listar archivos)

Archivos maestros: MAEDIR (mostrar path y listar archivos)

Directorio de arribo de archivos externos: ARRIDIR

Archivos externos aceptados: ACEPDIR

Archivos externos rechazados: RECHDIR

Archivos procesados: PROCDIR

Reportes de salida: REPODIR

Logs de auditoría del Sistema: LOGDIR/<comando>.LOGEXT

Estado de la instalación: COMPLETA

Proceso de Instalación Cancelado

* 1. Si falta instalar algún componente, mostrar y grabar en el log los siguientes mensajes:

TP SO7508 . Tema Copyright © Grupo xx

Librería del Sistema: CONFDIR (mostrar path y listar archivos)

Ejecutables: BINDIR (mostrar path y listar archivos)

Archivos maestros: MAEDIR (mostrar path y listar archivos)

Directorio de arribo de archivos externos: ARRIDIR

Archivos externos aceptados: ACEPDIR

Archivos externos rechazados: RECHDIR

Archivos procesados: PROCDIR

Reportes de salida: REPODIR

Logs de auditoria del Sistema: LOGDIR/<comando>.LOGEXT

Componentes faltantes: listado de los componentes faltantes

Estado de la instalación: INCOMPLETA

Desea completar la instalación? (Si-No)

* 1. Si el usuario indica Si:
     1. Chequear que Perl esté instalado
     2. Brindar las indicaciones para completar el proceso de Instalación indicando en qué lugar se instalarán los componentes (como se deben usar los que indica el archivo de configuración (archivo .conf), el usuario no podrá cambiar estos valores)
     3. Mostrar los valores de instalación

TP SO7508 . Tema Copyright © Grupo xx

Librería del Sistema: CONFDIR

Ejecutables: BINDIR

Archivos maestros: MAEDIR

Directorio de arribo de archivos externos: ARRIDIR

Espacio mínimo libre para arribos: DATASIZE Mb

Archivos externos aceptados: ACEPDIR

Archivos externos rechazados: RECHDIR

Archivos procesados: PROCDIR

Reportes de salida: REPODIR

Logs de auditoria del Sistema: LOGDIR/<comando>.LOGEXT

Tamaño máximo para los archivos de log del sistema: LOGSIZE Kb

Estado de la instalacion: LISTA

* + 1. Continuar en el paso: “Confirmar Inicio de Instalación”
  1. Si el usuario indica No, ir a **FIN**
  2. Si el paquete no fue instalado, continuar en el siguiente punto

###### Aceptación de términos y condiciones

Mostrar en consola el siguiente mensaje:

TP SO7508 . Tema Copyright © Grupo xx

A T E N C I O N: Al instalar TP SO7508 UD. expresa aceptar los términos y Condiciones del "ACUERDO DE LICENCIA DE SOFTWARE" incluido en este paquete.

Acepta? Si – No

* 1. Si el usuario indica No, ir a **FIN**
  2. Si el usuario indica Si, continuar
  3. Cualquier otra cosa, seguir en este punto

###### Chequear que Perl esté instalado

Para la instalación de TP SO7508 es necesario chequear que Perl versión 5 o superior este instalado.

* 1. Si al chequear esto da error, entonces mostrar y grabar en el log:

TP SO7508 . Tema Copyright © Grupo xx

Para instalar el TP es necesario contar con Perl 5 o superior instalado. Efectúe su instalación e inténtelo nuevamente.

Proceso de Instalación Cancelado

* + 1. ir a **FIN**
  1. Si Perl está instalado, mostrar y grabar en el log un mensaje informativo con la versión de perl que se encuentra instalada y continuar.

TP SO7508 . Tema Copyright © Grupo xx

Perl Version:

###### Definir el directorio de instalación de los ejecutables

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de instalación de los ejecutables ($grupo/bin):

Proponer /bin y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “bin” o un subdirectorio como /tp/sistemas/bin

Reservar este path en la variable BINDIR

###### Definir el directorio de instalación de los archivos maestros

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de instalación de los archivos maestros ($grupo/mae):

Proponer /mae y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “mae” o un subdirectorio como /data/mae

Reservar este path en la variable MAEDIR

###### Definir el directorio de arribo de archivos externos

Se debe solicitar al usuario que indique cual es el directorio que permite el arribo de archivos externos. Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de arribo de archivos externos ($grupo/arribos):

Proponer /arribos y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “/arribos” o un subdirectorio como /archivos/externos/arribos

Reservar este path en la variable ARRIDIR

###### Definir el espacio mínimo libre para el arribo de archivos externos

Se debe solicitar al usuario que indique cual es el espacio mínimo libre en el directorio ARRIDIR.

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el espacio mínimo libre para el arribo de archivos externos en Mbytes (100):

Proponer 100 Mb, Si el usuario lo desea cambiar, debe ingresar una cantidad que se interpreta como Mb. Reservar este valor en la variable DATASIZE.

###### Verificar espacio en disco

Chequear si en ARRIDIR hay disponibles por lo menos DATASIZE Mb. Si esto da error mostrar y grabar en el log el siguiente mensaje:

Insuficiente espacio en disco.

Espacio disponible: xx Mb.

Espacio requerido $DATASIZE Mb

Cancele la instalación e inténtelo mas tarde o vuelva a intentarlo con otro valor.

Volver a verificar espacio en disco.

###### Definir el directorio de grabación de los archivos rechazados

Se debe solicitar al usuario que indique cual es el directorio que desea usar para mover los archivos externos rechazados. Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de grabación de los archivos externos rechazados ($grupo/rechazados):

Proponer /rechazados y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “/rechazados” o un subdirectorio como /archivos/externos/rechazados

Reservar este path en la variable RECHDIR

###### Definir el directorio de grabación de los archivos aceptados

Se debe solicitar al usuario que indique cual es el directorio que desea usar para mover los archivos externos aceptados. Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de grabación de los archivos externos aceptados ($grupo/aceptados):

Proponer /aceptados y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “/aceptados” o un subdirectorio como /archivos/externos/ aceptados

Reservar este path en la variable ACEPDIR

###### Definir el directorio de trabajo principal del proceso

Se debe solicitar al usuario que indique cual es el directorio de trabajo principal que desea usar para mover los archivos procesados. Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de grabación de los archivos procesados ($grupo/procesados):

Proponer /procesados y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “/procesados” o un subdirectorio como /archivos/externos/ procesados

Reservar este path en la variable PROCDIR

###### Definir el directorio de grabación de los reportes de salida

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el directorio de grabación de los reportes de salida ($grupo/reportes):

Proponer /reportes y si el usuario lo desea cambiar, permitírselo.

El usuario puede ingresar un nombre simple como “reportes” o un subdirectorio como archivos/salida/reportes

Reservar este path en la variable REPODIR.

###### Definir el directorio de logs para los comandos

Se debe solicitar al usuario el directorio donde desea dejar los archivos de log del práctico, para ello mostrar en pantalla el siguiente mensaje:

Defina el directorio de logs ($grupo/log):

Si el usuario acepta el nombre propuesto, continuar

Si el usuario lo desea cambiar, debe ingresar un nombre de directorio

Reservar este path en la variable LOGDIR.

###### Definir la extensión para los archivos de log

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

"Ingrese la extensión para los archivos de log: (.log)"

Proponer .log y si el usuario lo desea cambiar, permitírselo.

Reservar ese valor en la variable LOGEXT.

###### Definir el tamaño máximo para los archivos de log

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Defina el tamaño máximo para los archivos $LOGEXT en Kbytes (400):

Proponer 400 KB, Si el usuario lo desea cambiar, debe ingresar una cantidad que se interpreta como Kb.

Reservar este valor en la variable LOGSIZE.

###### Mostrar estructura de directorios resultante y valores de parámetros configurados

* 1. Limpiar la pantalla
  2. Mostrar y grabar en el log el siguiente mensaje con su respuesta:

TP SO7508 . Tema Copyright © Grupo xx

Librería del Sistema: CONFDIR

Ejecutables: BINDIR

Archivos maestros: MAEDIR

Directorio de arribo de archivos externos: ARRIDIR

Espacio mínimo libre para arribos: DATASIZE Mb

Archivos externos aceptados: ACEPDIR

Archivos externos rechazados: RECHDIR

Archivos procesados: PROCDIR

Reportes de salida: REPODIR

Logs de auditoria del Sistema: LOGDIR/<comando>.LOGEXT

Tamaño máximo para los archivos de log del sistema: LOGSIZE Kb

Estado de la instalacion: LISTA

* 1. Si el usuario indica Si, Continuar en el paso: “Confirmar Inicio de Instalación”
  2. Si el usuario indica No, Limpiar la pantalla y continuar después del chequeo de Perl

En este caso, los valores default propuestos deben ser os recientemente ingresados por el instalador, es decir los contenidos en las variables: BINDIR, ARRIDIR, DATASIZE, LOGDIR, LOGEXT, LOGSIZE, etc

###### Confirmar Inicio de Instalación

Mostrar y grabar en el log el siguiente mensaje con su respuesta:

Iniciando Instalación. Esta Ud. seguro? (Si-No)

* 1. Si el usuario indica Si, Continuar en el paso: “Instalación”
  2. Si el usuario indica No, ir a FIN

###### Instalación

* 1. Crear las estructuras de directorio requeridas mostrando el siguiente mensaje:

Creando Estructuras de directorio. . . .

$BINDIR

$MAEDIR

$ARRIDIR

$RECHDIR

$ACEPDIR

$PROCDIR

$LOGDIR

$REPODIR

* 1. Mover los archivos maestros al directorio MAEDIR mostrando el siguiente mensaje

Instalando Archivos Maestros

* 1. Mover la tabla de separadores y la tabla de campos al directorio CONFDIR mostrando el siguiente mensaje

Instalando Tablas de Configuración

* 1. Mover los ejecutables y funciones al directorio BINDIR mostrando el siguiente mensaje

Instalando Programas y Funciones

* 1. Actualizar el archivo de configuración mostrando el siguiente mensaje

Actualizando la configuración del sistema

Se debe almacenar la información de configuración del sistema en el archivo .conf en CONFDIR

Si el archivo de configuración no existe, crearlo, si existe actualizar los valores que correspondan.

Se debe grabar un registro para cada una de las siguientes variables:

GRUPO, CONFDIR, BINDIR, MAEDIR, ARRIDIR, ACEPDIR, RECHDIR, PROCDIR, REPODIR, LOGDIR, LOGEXT, LOGSIZE, DATASIZE

###### Borrar archivos temporarios, si los hubiese generado

###### Mostrar mensaje de fin de instalación

Instalación concluida

###### FIN

Cerrar el archivo .log

Terminar el proceso

## Inicialización:

Descripción

El propósito de este comando es preparar el entorno de ejecución del TP (ambiente).

* Es el primero en orden de ejecución
* Se dispara manualmente
* Graba en el archivo de Log
* Invoca, si corresponde, al siguiente paso

El Proceso se inicia con el aseguramiento de la disponibilidad de la información para llevar adelante el proceso total: Es indispensable contar con los comandos, archivos maestros y las tablas de configuración con los permisos adecuados

A los efectos exclusivos de este tp solo se valida que estos archivos existan en el directorio correspondiente, si no tienen los permisos adecuados, los setea. Los comandos deben estar disponibles para ejecución y los archivos para su lectura pero no para su actualización.

Continúa con el seteo inicial de las variables de ambiente, incluyendo la variable PATH, para permitir la correcta ejecución de los scripts por parte de usuarios que no son root.

En el resto de los comandos, cuando necesitan referenciar a un directorio del sistema, deben hacerlo a través de las variables de ambiente que setea este script, evitando de esta manera el hardcode de directorios.

También configura valores particulares para la ejecución del comando y ofrece ejecutarlo.

Pasos Sugeridos:

###### 1-Inicializar el archivo de log

Comando Inicio de Ejecución

###### 2-Verificar que la instalación está completa y que los archivos tengan los permisos adecuados

Debe asegurar la disponibilidad de la información indispensable para la ejecución del sistema, como ser comandos, archivos maestros y las tablas de configuración (Tabla de Separadores y Tabla de Campos)

**S**i se detecta algún faltante en la instalación o algun permiso inadecuado, explicar la situación, indicar los componentes faltantes y terminar la ejecución.

###### 3- Verificar si el ambiente ya ha sido inicializado.

debe setear las variables de ambiente una sola vez por cada sesión de usuario.

Si el ambiente ya fue inicializado, no falta ningún componente ni permiso las variables de ambiente estan seteadas, etc y aun asi se intenta ejecutar nuevamente este comando, explicar la situación e ir al paso FINAL.

4- Si nunca fue ejecutado en esta sesión de usuario

Verificar que la instalación está completa y que los archivos tengan los permisos adecuados (idem paso 2)

Si se detecta algún faltante en la instalación, explicar la situación, indicar los componentes faltantes y terminar la ejecución.

Si se detecta que algún archivo no tiene los permisos adecuados, setarlos correctamente y continuar en el siguiente paso

###### 5-Setear la variable PATH y cualquier otra variable de ambiente que considere necesarias, como ser: GRUPO, ARRIDIR, RECHDIR, BINDIR, MAEDIR, REPODIR, LOGDIR LOGEXT, etc.

###### 6-Solicitar al usuario que ingrese las variables de configuración del comando

Se debe solicitar al usuario la cantidad de ciclos de (variable CANLOOP) y el tiempo de espera entre cada ciclo de (variable TESPERA), para ello mostrar en pantalla el siguiente mensaje:

"Cantidad de Ciclos de ? (100 ciclos)"

Almacenar este valor en CANLOOP

Validar que CANLOOP sea siempre positivo

"Tiempo de espera entre ciclos? (1 minuto)"

Almacenar este valor en TESPERA

Validar que TESPERA sea siempre mayor o igual a un minuto

Si el usuario ingresa un valor incorrecto volver a preguntar.

###### 7-Ver si se desea arrancar

debe ofrecer la posibilidad de arrancar el demonio por ejemplo mostrando el siguiente mensaje:

“Desea efectuar la activación de ?” Si – No

###### 7.1 Si el usuario no desea arrancar el demonio, entonces explicar cómo hacerlo manualmente

###### 7.2 Si el usuario desea arrancar el demonio, activarlo

###### FINAL:

Si el ambiente esta inicializado correctamente, **m**ostrar el contenido de las variables**, el process id del demonio (si es que esta corriendo)** y el estado del sistema**, por ejemplo:**

TP SO7508 . Tema Copyright © Grupo xx

Librería del Sistema: CONFDIR (mostrar path y listar archivos)

Ejecutables: BINDIR (mostrar path y listar archivos)

Archivos maestros: MAEDIR (mostrar path y listar archivos)

Directorio de arribo de archivos externos: ARRIDIR

Archivos externos aceptados: ACEPDIR

Archivos externos rechazados: RECHDIR

Archivos procesados: PROCDIR

Reportes de salida: REPODIR

Logs de auditoria del Sistema: LOGDIR/<comando>.LOGEXT

(listar todas las otras variables que deseen mostrar)

Estado del Sistema: INICIALIZADO

Demonio corriendo bajo el no.: <Process Id de >

###### Cerrar el archivo de log- Terminar el proceso

## Detección y Recepción:

Input

* Archivos (cualesquiera) que arriban al directorio $ARRIDIR/<nombre del archivo>
* Maestro de Países y Sistemas $MAEDIR/p-s.mae

Output

* Archivos Recibidos $ACEPDIR/<pais>-<sistema>-<año>-<mes>
* Archivos (cualesquiera) Rechazados $RECHDIR/<nombre del archivo>
* Log $LOGDIR/.$LOGEXT

Opciones y Parámetros

* A especificar por el desarrollador

Descripción

El propósito de este comando es detectar la llegada de archivos al directorio $ARRIDIR, efectuar la validación del nombre del archivo que detecta y ponerlo a disposición del siguiente paso. Si el archivo no es válido, debe rechazarlo. Debe arrancar, si corresponde, al Intérprete.

* Es el segundo en orden de ejecución
* Es un proceso del tipo “Demonio” :
  + Un demonio, daemon o dæmon (de sus siglas en inglés Disk And Execution Monitor), es un tipo especial de proceso informático que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario (es un proceso no interactivo). Este tipo de programas se ejecutan de forma continua.
* Este comando puede ser disparado manualmente o a través del comando
* Graba en el archivo de Log, solo cuando efectúa el movimiento de recepción o rechazo

Para determinar si un archivo es válido o no, se debe verificar su nombre. Los archivos válidos poseen nombres con este formato: <país>-<sistema>-<año>-<mes>

* <pais >, corresponde a un código de país
* <sistema>, corresponde a un código de sistema
* La combinación código de país + código de sistema debe existir en el maestro de Países y Sistemas (p-s.mae)
* <año>, corresponde a un año mayor a 2000. El formato del año es aaaa
* <mes>, corresponde al valor de un mes (entre 1 y 12), El formato del mes es mm.
* A la combinación año-mes se la denomina: PERIODO, el periodo no puede ser mayor al periodo corriente

Es decir que al directorio $ARRIDIR pueden arribar archivos provenientes de varios países, sistemas, años o meses. Ejemplos de nombres de archivos validos

A-6-2012-06

B-7-2012-08

A-6-2012-12

A-7-2012-06

Si en $ARRIDIR se reciben otro tipo de archivos, se los debe rechazar

El proceso arranca, contabiliza el ciclo, verifica el directorio $ARRIDIR, verifica los archivos, acepta o rechaza según corresponda, determina si debe terminar (CANLOOP) o dormir un tiempo x (TESPERA) para volver a empezar.

No se puede ejecutar este comando si la inicialización de ambiente no fue realizada o si existe otro demonio corriendo.

Pasos sugeridos:

1. Grabar en el Log el nro de ciclo: Ej: “Ciclo Nro 1”
2. Chequear si hay archivos en el directorio $ARRIDIR. Si existen archivos, por cada archivo que se detecta
   1. Verificar que el archivo sea un archivo común, de texto. Los archivos de cualquier otro tipo, se rechazan.
   2. Verificar que el formato del nombre del archivo sea correcto, los archivos con nombres que no se correspondan con el formato esperado, se rechazan.
   3. Si el nombre del archivo es válido mover el archivo aceptado a $ACEPDIR empleando la función y grabar en el log el mensaje de éxito
   4. Si el nombre del archivo no es válido mover el archivo rechazado a $RECHDIR empleando la función , grabar en el log el mensaje de rechazo aclarando cual es el motivo:
      * + Tipo de archivo invalido
        + nombre del archivo con formato invalido
        + País inexistente
        + Sistema inexistente
        + Año invalido
        + Mes invalido
        + Periodo mayor al periodo corriente
        + cualquier otro error que considere pertinente indicar.
3. Una vez que se hayan procesado todos los archivos que existen en $ARRIDIR se debe chequear la existencia de archivos en el directorio $ACEPDIR (ya sean del ciclo actual o de ciclos anteriores).
4. Si existen archivos en $ACEPDIR
   1. Invocar al Comando siempre que éste no se esté ejecutando.

Si arranca correctamente se debe mostrar por pantalla el process id de

Si da algún tipo de error se debe mostrar por pantalla el mensaje explicativo

1. Si aún no alcanzó la cantidad de ciclos definida en la variable de ambiente en CANLOOP Dormir un tiempo TESPERA y Volver al punto 1
2. Si se alcanzó la cantidad de ciclos definida en la variable de ambiente CANLOOP, terminar

## 

Input

* Archivos Recibidos $ACEPDIR/<pais>-<sistema>-<año>-<mes>
* Tabla de Separadores $CONFDIR/**T1.tab**
* Tabla de Campos $CONFDIR/**T2.tab**

Output

* Archivos de Préstamos Personales por país $PROCDIR/**PRESTAMOS.<pais>**
* Archivos (duplicados) Rechazados $RECHDIR/<nombre del archivo>
* Archivos Procesados $PROCDIR/<pais>-<sistema>-<año>-<mes>
* Log $LOGDIR/**.**$LOGEXT

Opciones y Parámetros

* A especificar por el desarrollador

Descripción

El propósito de este comando es leer los archivos que se encuentran en el directorio $ACEPDIR y grabar sus registros formateados en archivos de préstamos por país.

* Es el tercero en orden de ejecución
* Este comando puede ser disparado manualmente o a través del comando
* Graba en su archivo de Log

Los archivos aceptados se encuentran en el directorio $ACEPDIR y contienen información de préstamos personales impagos generada por los diferentes sistemas de las subsidiarias que operan en distintos países.

Es por ello que no poseen una estructura estándar. La misma se encuentra definida a través de las tablas de separadores y campos que tiene el sistema. Para realizar la lectura de los archivos se debe interpretar su información a través de estas tablas y generar una estructura estándar que pueda ser usada para las consultas y alineada a los formatos del archivo maestro contable de Préstamos Personales Impagos:

* Usa como separador decimal siempre la coma
* Usa como formato de fecha siempre dd/mm/aaaa
* Usa como separador de campos el punto y coma

No se debe procesar dos veces un mismo archivo

No se puede ejecutar este comando si la inicialización de ambiente no fue realizada o si existe otro interprete corriendo.

Pasos sugeridos

###### Inicializar el Log

Inicializar el log grabando: Inicio de Interprete y Cantidad de Archivos de input

1. **Procesar Un Archivo aceptado**

Los archivos de input se encuentran en $ACEPDIR

1. **Verificar que no sea un archivo duplicado**

Como los archivos ya interpretados se movieron oportunamente (y tal cual fueron recibidos) a $PROCDIR, es posible detectar tempranamente si el archivo vino duplicado analizando este directorio.

1. Si en ese directorio ya existe un archivo de igual nombre, entonces el archivo a procesar se lo considerará duplicado. En este caso, grabar en el archivo de log “DUPLICADO: <nombre del archivo>“ y mover el archivo a $RECHDIR empleando la función .
2. Seguir con el siguiente archivo aceptado
3. **Determinar Separadores**

Se deben recuperar de la tabla T1 los valores de separador de campo y separador decimal para el país y sistema correspondiente.

La tabla de Separadores o Tabla T1 nos dice cual es el separador de campo y cuál es el separador decimal que usa el archivo de input según el país y el sistema de origen.

1. **Determinar Campos**

Se debe recuperar de la tabla de Campos o Tabla T2 la estructura del registro de input y el formato de los campos que nos interesan leer según el país y el sistema de origen.

La tabla de Campos o Tabla T2 nos dice que campos, en qué orden de aparición vamos a encontrar esos campos en los archivos de préstamos personales recibidos y con qué formato viene informado.

El formato de los campos de fecha:

* + El formato de las fechas puede ser: ddmmyy<longitud>. O yymmdd<longitud>.

Donde dd corresponde a dos caracteres para el dia, mm corresponde a dos caracteres para el mes y yy corresponde a cuatro caracteres para el año. La interpretación de la longitud es la siguiente: Si la longitud es 8, no posee separador entre día, mes y año, si la longitud es 10, posee separador entre día, mes y año. El separador de fecha lo desconocemos. Solo podemos asegurar que nunca es igual al separador de campos ni al separador decimal.

El formato de los campos alfanuméricos

* + El formato de los alfanuméricos es: $<longitud>.

El $ nos indica que el formato es alfanumérico y <longitud> indica la longitud máxima del campo.

El formato de los campos numéricos

* + El formato de los numéricos es: commax<longitud parte entera>.<longitud parte decimal>

El commax nos indica que el formato es numérico, <longitud parte entera> nos indica la longitud máxima de la parte entera, <longitud parte decimal > nos indica la longitud máxima de la parte decimal.

###### Por cada Registro del Archivo a Interpretar

###### Leer el registro

Se debe leer el registro de input interpretando la estructura de acuerdo a los separadores obtenidos en 4 y los campos determinados en 5

###### Construir el registro de préstamo

Interpretar la fecha CTB\_FE y determinar dia mes y año a grabar

Interpretar los montos y llevarlos al formato estándar para grabarlos. Todos los campos de montos deben ser grabados en cero si no están informados

Cálculo de monto restante

El campo de monto restante debe ser calculado como la suma de los montos PRES, IMPAGO, INDE, INNODE y deducir el monto debitado DEB

MT\_REST = MT\_PRES + MT\_IMPAGO + MT\_INDE + MT\_INNODE – MT\_DEB

Los registros con monto restante menor a cero no se graban.

###### Grabar, si corresponde, el registro de salida

Grabar el registro en el archivo $PROCDIR**/PRESTAMOS.<país>** correspondiente

Si el Archivo ya existe, agregarle los nuevos registros. Si no existe, crearlo.

Repetir el paso 6 hasta que se termine el archivo.

###### Fin de Archivo

Para evitar el reprocesamiento de un mismo archivo, mover el archivo procesado a: $PROCDIR empleando la función .

Grabar en el Log la cantidad de Registros de Input y la cantidad de Registros de Output

Repetir desde el paso 3 hasta que se terminen todos los archivos.

###### Cerrar el Log

Cerrar el log grabando “Fin de ”

## Consultas y Reportes:

Input

* Maestro de Países y Sistemas $MAEDIR/**p-s.mae**
* Archivo Maestro Contable de Préstamos Personales Impagos $MAEDIR/**PPI.mae**
* Archivos de Préstamos Personales por país $PROCDIR/**PRESTAMOS.<pais>**

Output

* Archivos de Recalculo $REPODIR/**RECALCULO.<pais>**
* Listados $REPODIR/.xxx

Opciones y Parámetros

* -a
* -g
* País (nombre del país)
* Sistema (código del sistema)
* Año
* Periodo (Año-mes)
* Rango de periodos
* Otros parámetros según la consulta

Descripción

Este comando se dispara manualmente, No se puede ejecutar si la inicialización de ambiente no fue realizada y si ya existe otro comando en ejecución

El propósito de este comando es resolver, mostrar y, eventualmente, grabar consultas vinculadas a la comparación entre dos fuentes: El archivo de préstamos por país (prestamos.<país>) y el archivo maestro (PPI.mae)

* + Es el cuarto en orden de ejecución
  + Se dispara manualmente
  + No Graba en el archivo de Log

Ambos archivos contienen información de contratos de préstamos personales impagos, una fuente (prestamos.<país>) es generada por los sistemas que operan en distintos países. La otra fuente (PPI.mae) es generada por el Sistema Contable Central.

El objetivo es desarrollar un comando de consultas para que el usuario pueda comparar y decidir si genera los archivos de recalculo

Requisitos:

* Se deben emplear estructuras Hash en la resolución (requisito indispensable)
* Debe presentar un menú amigable y una **opción (-a)** de ayuda del comando
* Debe permitir al usuario efectuar N consultas sin salir del comando
* Debe mostrar los resultados por pantalla indicando en el encabezado los datos de la consulta
* Debe permitir grabar un reporte de salida con lo mismo que se muestra por pantalla **(opción -g)**. El nombre del reporte debe ser $/REPODIR/.**xxx** dónde xxx es un descriptor siempre distinto que asegura no sobrescribir ningún informe previo.
* Debe solicitar al usuario un nombre de país para seleccionar el archivo de presupuesto a usar en la consulta (siempre procesa un solo país, si quiere procesar otro país, debe invocar nuevamente)
* parámetros opcionales
  + Debe permitir filtrar por sistema
  + Debe permitir filtrar año
  + Debe permitir filtrar por periodo (año-mes) o rango de periodos (pe. desde septiembre hasta noviembre del 2012 o desde octubre del 2012 hasta febrero del 2013)
* Si se desea hacer una nueva consulta debe permitir cambiar los filtros o cambiar de pregunta

Consultas a resolver:

1. **Comparación para el recalculo**

Primer PASO:

Según los filtros ingresados, determinar los registros DEL MAESTRO que se van a comparar

|  |  |
| --- | --- |
| Código de País del maestro | código de país pasado como parámetro |
| Código de Sistema del maestro | Código de Sistema pasado como parámetro o todos, según corresponda |
| Año contable del maestro | Año contable pasado como parámetro o todos, según corresponda |
| Mes contable del maestro | mes contable pasado como parámetro o todos, según corresponda |

Segundo PASO:

Calcular el monto restante del maestro

* El monto restante del maestro debe ser calculado como la suma de los montos PRES, IMPAGO, INDE, INNODE y deducir el monto debitado DEB

Tercer PASO:

El registro de prestamos.país contra el cual se efectúa la comparación es aquel donde:

|  |  |
| --- | --- |
| Código Préstamo del maestro | = Código Préstamo de prestamos.pais |
| Año contable del maestro | = Año contable de prestamos.pais |
| Mes contable del maestro | = Mes contable de prestamos.pais |

Si no se halla registro en prestamos.pais, no se hace la comparación

Si hay mas de un registro contra el cual comparar, tomar el de mayor día contable en prestamos.pais

SI AÚN ASI hay más de un registro contra el cual comparar, tomar el de fecha de grabación mas reciente

Cuarto PASO:

Determinar el valor de la “recomendación” que puede ser “RECALCULO” o “BUENO”. La recomendación se determina de la siguiente manera

* Si estado contable del maestro = SMOR y estado contable prestamos.pais distinto a SMOR, recomendación = RECALCULO
* Si monto restante del maestro menor a monto restante de prestamos.pais, recomendación = RECALCULO

Mostrar los resultados con un titulo representativo de la consulta y filtros usados

|  |
| --- |
| Código Préstamo |
| Código del cliente |
| Estado contable del maestro |
| Estado contable del “prestamos.pais” |
| Monto restante del maestro |
| Monto restante del “prestamos.pais” |
| Recomendacion |

Mostrar el total de registros listados

Quinto PASO:

Preguntar si se desea grabar RECALCULO y grabarlo si corresponde

El nombre del archivo de recalculo es $REPODIR/**RECALCULO.<pais>.** Si el archivo no existe, crearlo. Si existe agregar los nuevos registros.

Sexto PASO:

Preguntar si se desea hacer otra consulta

1. **Diferencia (en valor absoluto) mayor al x%**

Para esta consulta se debe solicitar que se ingrese un porcentaje de comparación X

Esta consulta responde a la pregunta: En qué casos la diferencia (en valor absoluto) entre el monto restante del maestro y el monto restante del país es mayor al X %.

Pasos 1,2,3 idem consulta anterior

Cuarto PASO:

Mostrar los resultados con un titulo representativo de la consulta y filtros usados

|  |
| --- |
| Código Préstamo |
| Monto restante del maestro |
| Monto restante del “prestamos.pais” |
| Diferencia |

Mostrar el total de registros listados. Ejemplo: X= 7%

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| préstamo | maestro | país | diferencia |  |
| 1 | 1000 | 700 | 30% | se lista |
| 2 | 1000 | 750 | 25% | se lista |
| 3 | 1000 | 800 | 20% | se lista |
| 4 | 1000 | 850 | 15% | se lista |
| 5 | 1000 | 900 | 10% | se lista |
| 6 | 1000 | 950 | 5% | no se lista |
| 7 | 1000 | 1000 | 0% | no se lista |
| 8 | 1000 | 1050 | -5% | no se lista |
| 9 | 1000 | 1100 | -10% | se lista |
| 10 | 1000 | 1150 | -15% | se lista |
| 11 | 1000 | 1200 | -20% | se lista |
| 12 | 1000 | 1250 | -25% | se lista |
| 13 | 1000 | 1300 | -30% | se lista |

Trece registros.

Quinto PASO, no aplica

Sexto PASO, idem consulta anterior

1. **Diferencia (en valor absoluto) mayor al x$**

Para esta consulta se debe solicitar que se ingrese un monto de comparación X

Esta consulta responde a la pregunta: En qué casos la diferencia (en valor absoluto) entre el monto restante del maestro y el monto restante del país es mayor a X $.

Pasos 1,2,3 idem consulta anterior

Cuarto PASO:

Mostrar los resultados con un titulo representativo de la consulta y filtros usados

|  |
| --- |
| Código Préstamo |
| Monto restante del maestro |
| Monto restante del “prestamos.pais” |
| Diferencia |

Mostrar el total de registros listados. Ejemplo: X= 70$

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| préstamo | maestro | país | diferencia |  |
| 1 | $ 1.000,00 | $ 700,00 | $ 300,00 | se lista |
| 2 | $ 1.000,00 | $ 750,00 | $ 250,00 | se lista |
| 3 | $ 1.000,00 | $ 800,00 | $ 200,00 | se lista |
| 4 | $ 1.000,00 | $ 850,00 | $ 150,00 | se lista |
| 5 | $ 1.000,00 | $ 900,00 | $ 100,00 | se lista |
| 6 | $ 1.000,00 | $ 950,00 | $ 50,00 | no se lista |
| 7 | $ 1.000,00 | $ 1.000,00 | $ 0 | no se lista |
| 8 | $ 1.000,00 | $ 1.050,00 | $ -50,00 | no se lista |
| 9 | $ 1.000,00 | $ 1.100,00 | $ -100,00 | se lista |
| 10 | $ 1.000,00 | $ 1.150,00 | $ -150,00 | se lista |
| 11 | $ 1.000,00 | $ 1.200,00 | $ -200,00 | se lista |
| 12 | $ 1.000,00 | $ 1.250,00 | $ -250,00 | se lista |
| 13 | $ 1.000,00 | $ 1.300,00 | $ -300,00 | se lista |
| 13 registros | |  |  |  |

Quinto PASO, no aplica

Sexto PASO, idem consulta anterior

# Especificación de Funciones

A continuación se brinda una descripción de cada función solicitada.

En algunos casos se sugieren los pasos de ejecución solo a los efectos de ordenar la explicación, por lo cual deben considerarse meramente indicativos.

Si el equipo de desarrollo lo considera pertinente, puede modificarlos tanto sea en el orden de ejecución como en la forma de resolverlo, siempre y cuando esto no afecte el resultado final esperado y lo aclare debidamente en las hipótesis.

## StartX

Opciones y Parámetros

* parámetros u opciones a especificar por el desarrollador

Descripción

Esta función tiene por objeto disparar procesos. Es complementaria a .

Puede ser invocada desde la línea de comando o bien desde otro comando.

No se puede arrancar un proceso si éste ya se encuentra corriendo. No se puede arrancar un proceso si la inicialización de ambiente no fue realizada. Indicar en las Hipótesis Globales donde se realizan estos controles

1. en forma centralizada en esta función
2. el control se delega al comando llamador

Si esta función es invocada por un comando que graba en un archivo de log, actualizarlo indicando el resultado de la operación

Explicar claramente su invocación y uso en el README.

Opciones y Parámetros

* parámetros u opciones a especificar por el desarrollador

Descripción

Esta función tiene por objeto detener procesos. Es complementaria a

Puede ser invocada desde la línea de comando o bien desde otro comando.

Si esta función es invocada por un comando que graba en un archivo de log, actualizarlo indicando el resultado de la operación

Explicar claramente su invocación y uso en el README.

Opciones y Parámetros

* Parámetro 1 (obligatorio): origen
* Parámetro 2 (obligatorio): destino
* Parámetro 3 (opcional): comando que la invoca
* Otros parámetros u opciones a especificar por el desarrollador

Descripción

Esta función tiene por objeto mover un archivo de un directorio a otro contemplando la posibilidad de archivos duplicados.

Puede ser invocada desde la línea de comando o bien desde otro comando.

En caso de archivos duplicados se debe gestionar un numero de secuencia nnn.

Este número de secuencia puede ser centralizado (una única secuencia para todo el sistema) o descentralizado (diferentes secuencias).

EVITE crear un archivo auxiliar para registrar el número de secuencia, si su implementación lo requiere, utilice el archivo de configuración y documente esto en las hipótesis generales del sistema.

Si esta función es invocada por un comando que graba en un archivo de log, actualizarlo indicando el resultado de la operación

debe devolver un código de retorno cero (0) si fue exitoso el movimiento o distinto de cero si tuvo errores.

Pasos Sugeridos

1. Antes de efectuar el movimiento, debe:
   1. Verificar si el origen y el destino son iguales. Si este fuera el caso, no mover
   2. Si el origen no existe o el destino no existe, es un error. Si este fuera el caso, no mover
   3. Verificar si es un archivo duplicado, es decir, si en el destino ya existe un archivo con ese mismo nombre. En este caso debe:
      1. Ver si existe dentro del directorio destino un subdirectorio /dup
         1. si no existe, crearlo
      2. Mover el archivo a ese subdirectorio con el siguiente nombre: <nombre del archivo original>.nnn dónde nnn es un número de secuencia que evita nombres duplicados.
      3. Eliminar el archivo del directorio de origen

Ejemplo mover arch.txt de /data1 a /data2

|  |  |
| --- | --- |
| Antes de MOVER | Luego de MOVER |
| /data1/arch.txt  /data2/arch.txt | /data1  /data2/arch.txt  /data2/dup/arch.txt.1 |

* 1. si no es un archivo duplicado, entonces hacer el move al destino indicado.

Ejemplo mover arch.txt de /data1 a /data2

|  |  |
| --- | --- |
| Antes de MOVER | Luego de MOVER |
| /data1/arch.txt  /data2 | /data1  /data2/arch.txt |

Opciones y Parámetros

* Parámetro 1 (obligatorio): comando
* Parámetro 2 (opcional): tipo de mensaje
* Parámetro 3 (obligatorio): mensaje
* Otros parámetros u opciones a especificar por el desarrollador

¿Qué es un Log?

Un log es un registro oficial de eventos durante un periodo de tiempo en particular.

Es usado para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (who, what, when, where y why, también llamado Standard W5) un evento ocurre para un dispositivo en particular o aplicación.

También se le considera como aquel mensaje que genera el programador de un sistema operativo, alguna aplicación o algún proceso, en virtud del cual se muestra un evento del sistema.

Es empleado por los profesionales de IT, auditoria y seguridad informática

¿Cómo debe ser un Log?

La escritura en el archivo de Log debe ser homogénea para todos los comandos por lo tanto debe estar centralizada a través de una función de librería.

Cada registro de log debe responder al estándar W5.

El Archivo de Log no debe crecer indiscriminadamente, debe preverse la forma de generar alertas o evitar su crecimiento sin control.

Para hacer amigable la lectura del archivo de log, en este TP se propone la función de visualización de logs:

Si el desarrollador lo desea, puede estar embebida dentro de esta función.

Descripción

Esta función tiene por objeto grabar en el log correspondiente los mensajes pasados como parámetro contemplando la creación del archivo de log cuando este no existiera.

Puede ser invocada desde la línea de comando o bien desde otro comando.

El nombre del Archivo de Log se indica en cada comando, por lo general coincide con éste.

La extensión del Archivo de Log se indica en la variable de ambiente $LOGEXT

El directorio donde se graba el log se indica en la variable de ambiente $LOGDIR (excepto para el log de la instalación que se graba en el directorio CONFDIR)

Si el archivo de log no existe, se debe crear. Si existe se le deben agregar los nuevos registros.

El tamaño máximo que puede alcanzar un archivo de log se indica en la variable de ambiente $LOGSIZE. Este valor se solicita en la instalación y debe ser empleado para evitar el crecimiento indiscriminado de los archivos de log.

Este tamaño máximo es un valor de referencia, en la práctica es posible encontrar archivos de log que superan el valor $maxlog dado que depende del momento en que se efectúa este control.

La idea es que cuando un archivo de log supera ese tamaño, se debe truncar el archivo dejando por ejemplo las últimas 50 líneas (valor arbitrariamente establecido). Si desea proponer otro mecanismo puede hacerlo aclarándolo debidamente en las hipótesis generales del sistema.

Cada vez que se reduce el tamaño del log, grabar en el log el mensaje: Log Excedido

Esto debe ser aplicado a todos los logs de comandos excepto al log de la instalación, este log nunca debe ser recortado.

Cada línea de log debe responder al estándar W5 (when, who, where, what and why):

¿Cuándo?

Fecha y Hora, en el formato que deseen y calculada justo antes de la grabación.

¿Quién?

Usuario, es el login del usuario

¿Dónde?

Comando, nombre del comando o función que genera el mensaje.

Se apreciará la utilidad de este parámetro cuando la función deba generar mensajes de log

¿Qué?

Tipo de Mensaje: Existen cuatro tipos de mensajes:

* I = INFORMATIVO: mensajes explicativos sobre el curso de ejecución del comando. Ej: Inicio de Ejecución
* W = WARNING (de alerta): mensajes de advertencia pero que no afectan la continuidad de ejecución del comando. Ej: Archivo duplicado
* E = ERROR: mensajes de error Ej: Archivo Inexistente.
* SE = SEVERAL ERROR: mensajes severos de error que provocarán la cancelación del comando: Ej: Archivo Maestro no encontrado

¿Por qué?

Mensaje, string de hasta 120 caracteres. Si cuentan con una tabla de mensajes pueden grabar código de mensaje y mensaje.

Todo el mensaje debe registrarse en una sola línea, el separador de registros debe ser new line.

ATENCION: Como el comando de instalación también debe grabar un log respetando la forma de escritura aquí indicada, aclare en las hipótesis generales del sistema que solución adoptaron para evitar duplicar este código

Opciones y Parámetros

* Parámetro 1 (obligatorio): comando
* Parámetro 2 (Filtros (opcionales y a definir por el desarrollador)
* Otros parámetros u opciones a especificar por el desarrollador

Descripción

Esta función tiene por objeto generar una visualización amigable del contenido de un archivo de log correspondiente al comando pasado como parámetro.

Se invoca desde la línea de comando.

Puede ser una función independiente o estar embebida dentro de la función empleando una combinación de opciones y parámetros en forma adecuada

Es deseable poder filtrar la visualización usando por ejemplo

* Opción –n donde n es una cantidad que sirve para indicar que se quieren ver las ultimas n líneas (-1 muestra la última línea, -10 muestra las últimas 10 líneas)
* Un filtro por string para que solo se muestren las líneas que contienen el string buscado
* Cualquier otro concepto de filtro o búsqueda definido por el desarrollador

Debido al alto grado de libertad que se les permite en el desarrollo de este comando, documentarlo detalladamente.

**FIN**