DCSM

(Distributable Centralized Scripts Manager)

Autor: Abdul Pallarés Calvi
e-mail: abdul@riereta.net

Objetivos de este documento

Detallar el uso de la plataforma así como su estructura final

Situación Actual

La situación general, en gran cantidad de redes, a la hora de gestionar los scripts de los servidores, que siempre existen, es crear los scripts específicos para cada servidor, y si es necesario en otro servidor se copia. Este sistema de administración de scripts es muy típico y, aunque evita algo de trabajo, tiene unas claras desventajas, las que intento solucionar son las siguientes:

1. Es difícil hacer un seguimiento de los scripts que están en cada servidor.
2. Si se modifica un script que se usa en más de un servidor hay que modificarlo en cada uno de los servidores, si tenemos en cuenta el punto 1 puede ser que no se apliquen los cambios deseados a todos los servidores.

Estas desventajas se pueden resolver mediante la centralización de la gestión y distribución de los scripts.

Situación final

Tras el desarrollo e implantación de la plataforma descrita en este documento tan solo se deberán crear o modificar los scripts en una sola máquina, que posteriormente distribuirá los scripts a los servidores necesarios, es decir, se crea un repositorio de scripts para toda la red. Al estar todos los scripts en una sola máquina, saber los scripts que se ejecutan en un servidor es una tarea tan sencilla como listar un directorio en la máquina que hospeda el repositorio, añadir los scripts necesarios a un nuevo servidor tan simple como conectarlo a la plataforma y modificar scripts genéricos en todos los servidores tan solo modificando el script en el repositorio y esperar a que se replique ;).

1. Visión general de la plataforma

Ya he explicado los objetivos deseados, así que en este apartado explicaré la lógica del proceso.

Como podréis ver la plataforma no tiene nada especial, se basa en un servidor de scripts o repositorio de scripts, este servidor almacenará los scripts en 2 direcotorios:

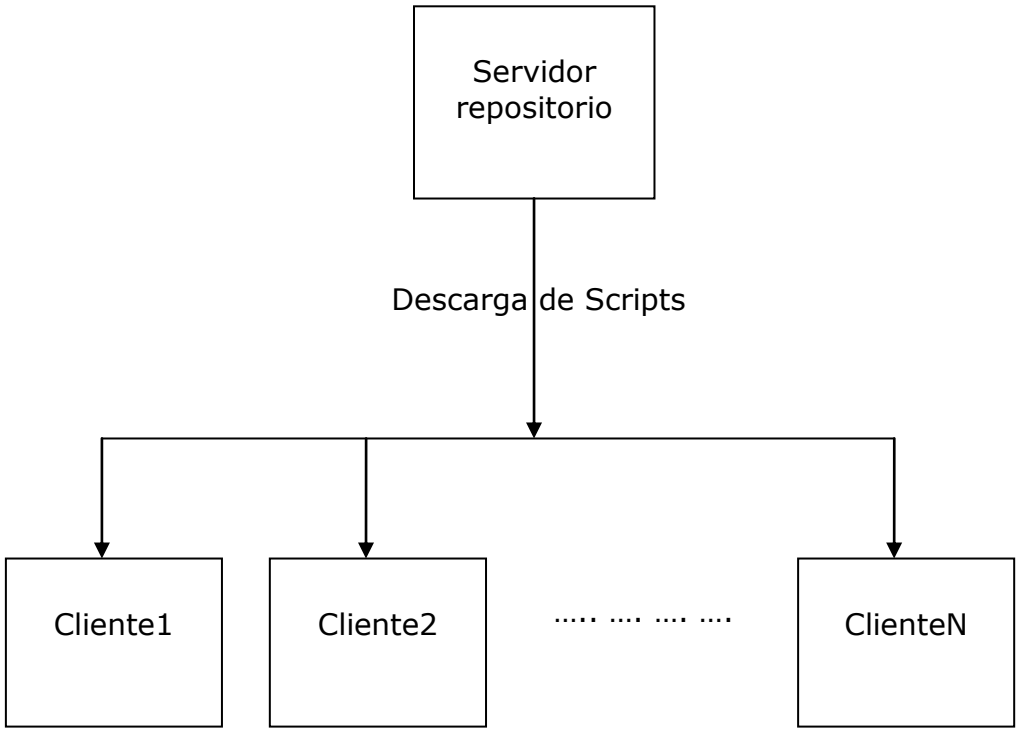
generic: Almacena los scripts que se definan como útiles en cualquier sistema de la red.

***hostname_cliente*:** Almacena scripts específicos de cada máquina.

El servidor de repositorio es el encargado de realizar el seguimiento de los cambios realizados a los scripts así como de los nuevos scripts añadidos. Los servidores que formen parte de la plataforma (Clientes) deberán comprobar periódicamente si existen cambios que deban actualizar.

Para facilitar la tarea de agregar nuevos clientes a la plataforma, el sistema incluye varios scripts de instalación que realizan los pasos necesarios para adecuar el entorno del cliente, creando los directorios, archivos, variables de entorno y tareas cron necesarias, una vez adecuado el entorno, descarga y ejecuta el script responsable de realizar las actualizaciones/instalaciones pertinentes, este será el script que se ejecute periódicamente para comprobar las actualizaciones.

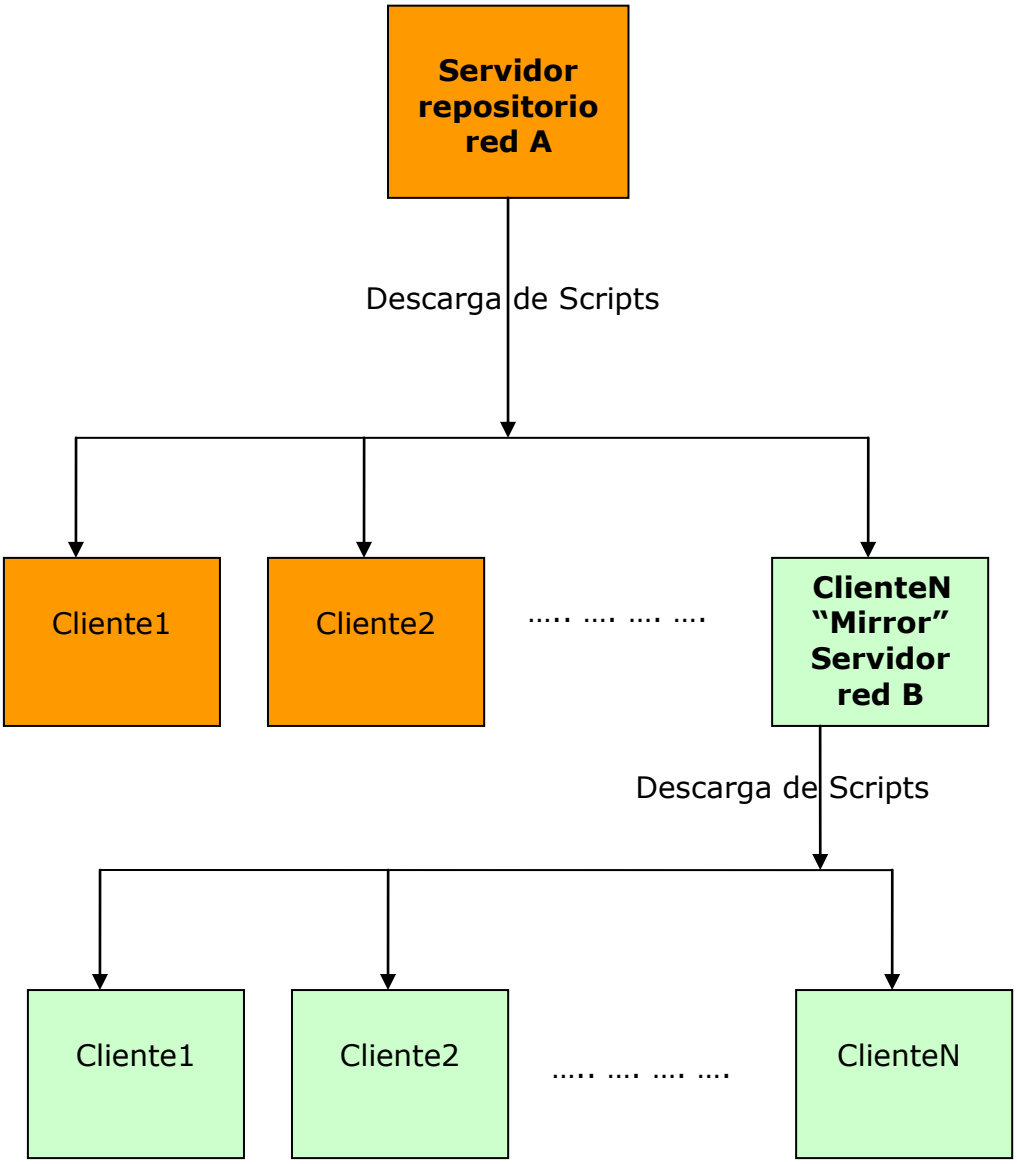
1.1. Esquema de la plataforma



Este es el esquema más simple de la plataforma, pero ¿qué sucede si la red está segmentada? Como es el caso que nos ocupa, bueno como el nombre de la plataforma indica, es una plataforma distribuible, ante este problema se podría permitir el acceso de cada cliente al servidor de repositorio, pero

es una solución pobre que rompe el sentido de la segmentación, sin embargo permitir el acceso de una sola máquina no rompe tanto el sentido de la segmentación. De esta forma se puede hacer un "mirror" del repositorio en la porción de red inaccesible, para hacer esto basta con crear un cliente que funcionará en un modo especial ya que descargará todos los scripts existentes en repositorio principal sin importar a qué máquina pertenezcan. Efectivamente esto se puede optimizar evitando que se descargue los scripts de los clientes de otras redes, pero la redundancia será mínima y ya sabemos que siempre va bien tener copias, en caso de ser necesario, los cambios para implementar esto no son muy complejos.

El esquema de una red segmentada simple es este



2. Detalle de la plataforma

2.1 Servidor

El servidor es básicamente un servidor CVS apoyado por unos scripts que facilitan la gestión de los módulos del repositorio, los scripts se encuentran en el directorio */scripts/*`*hostname*` y el directorio raíz del CVS en */var/lib/cvsd/scripts/* del servidor. Los scripts de específicos del servidor son los siguientes:

- **addmodule.sh**: Crea un nuevo módulo en el servidor CVS, cada cliente que tenga scripts exclusivos para sí mismo, debe tener un módulo que los contenga y cuyo nombre coincida con el hostname del cliente. Recibe como parámetro el nombre del módulo a crear.
- **removemodule.sh**: Elimina el módulo especificado del servidor CVS, así como el directorio que lo contiene bajo */scripts*. Antes de eliminar los archivos físicos, realiza una copia de seguridad con el siguiente formato */scripts/modulename.backup_dia-mes-año*. Recibe como parámetro el nombre del módulo a eliminar.
- **removefiles.sh**: Elimina archivos de un módulo y físicamente, antes de eliminar los archivos físicos realiza un backup en el directorio */scripts/modulename.backup_dia-mes-año*. Sus parámetros son: el nombre del módulo del que se desea eliminar, seguido del nombre de los archivos a eliminar separados por espacio.
- **updatemodule.sh**: Actualiza el módulo especificado, ya sea actualizando cambios en los scripts o agregando nuevos scripts al módulo. Recibe el nombre del módulo a actualizar.
- **listmodule.sh**: Este script lista los archivos contenidos en el modulo especificado. Su parámetro es el nombre del módulo a listar.
- **installclient.sh**: Este script sirve para configurar todo lo necesario para que una máquina pueda usar la plataforma y realizar la primera descarga de los módulos desde el repositorio. Tras su ejecución el cliente queda configurado para actualizarse a diario.

***Nota*:** Todos los scripts muestran un mensaje sobre su uso (parámetros requeridos y su orden) si se ejecutan sin ningún parámetro.

2.2 Cliente

El cliente consta básicamente de los dos archivos siguientes:

- **global.conf**: Este es el archivo de configuración general, en él se definen las variables generales a todos los scripts, actualmente tan solo contiene el directorio base de la plataforma y los datos de conexión al repositorio cvs:
  - SCRIPTS=/scripts
  - PSERVER=:pserver:anonymous@servidor:/scripts
- **updater.sh**: Este es el script principal de la plataforma, se encarga de conectar al servidor, descargar las actualizaciones y buscar y realizar las actualizaciones del crontab necesarias tras la actualización, este script incluye a "*global.conf*" y en caso de no encontrarlo mostrará un mensaje de error.
- **default_action**: Este archivo tan solo indica la acción por defecto a la hora de realizar algunas acciones con cvs. Debe contener exclusivamente la palabra: *continue*.

Nota: El archivo de configuración "*global.conf*" es utilizado por todos los scripts que requieran una entrada en *crontab* para su funcionamiento, y

aunque contenga pocas variables es imprescindible para el correcto funcionamiento de la plataforma.

3. Formato de los archivos script distribuidos

En general los scripts empezarán con un comentario que explique la función del mismo, posteriormente incluyen el archivo de configuración global, si no existe y lo requieren deben salir con "*exit 1*", tras la inclusión del archivo global, se deben definir las variables específicas (o locales) del script, seguidas de la inclusión del archivo de configuración local "*$0.conf*" que, en caso de existir, define una configuración especifica para un script en el cliente en el que se ejecute, así se pueden tener scripts genéricos con configuraciones especificas. El archivo de configuración local, por lo general sobrescribirá las variables de la sección de variables locales del script que lo incluye. Los scripts deben incluir en una línea no comentada, la orden cron a ejecutar por el sistema, el proceso de instalación de scripts leerá esta línea y la agregará o sustituirá en el crontab, aunque esta línea puede estar en cualquier parte del script recomiendo que se añada al inicio del bloque de las variables locales. El formato de la línea para un cron es *CRONDEF="linea del crontab"*. Veamos como ejemplo la cabecera del propio motor de la plataforma, la del script "*updater.sh*":

```
#!/bin/sh
            ############################
            #      Script to update client's modules    #
            #          from the scripts repository        #
            ############################

                  #### Global variables ###
if [ -f ../generic/global.conf ]; then
      . ../generic/global.conf
elif [ -f ./generic/global.conf ]; then
      . ./generic/global.conf
else
      echo "ERROR: generic/global.conf doesn't exists. Run
\"cvs:pserver:anonymous@servidor:/scripts checkout generic\" in the
cripts folder to solve the problem."
      exit 1
fi
                  #### Local variables ###
#Format del cron: minuts(0-59) hores(0-23) "dia del mes"(0-31) mes(1-
12) "dia de la setmana"(0-7, 0 i 7 son diumenge)
#For disable the automatic adition of this script into cron comment this
line
CRONDEF="00 1 * * * sh $SCRIPTS/generic/updater.sh"

hostname=`hostname`
MAXBACKUPS=5
if [ -d /tmp ]; then
      TMPDIR=/tmp
elif [ -d /var/tmp ]; then
      TMPDIR=/var/tmp
else
```

```
        TMPDIR=/
  fi
    ### Include local variables' file.It can  overwrite all previous ones ###
  if [ -f $0.conf ];then
        . $0.conf
  fi
```

Como se puede observar la "cabecera" del script está separada por secciones separadas por comentarios con el nombre de la sección. En la siguiente sección se verán unos ejemplos de uso de la plataforma.

4. **Ejemplo de uso práctico**

Convenciones para este ejemplo:

Hostname del servidor = *server1*

Hostname del cliente = *cliente1*

Variable global = variable definida en el archivo global.conf y que es compartida por varios scripts de la plataforma

Variable local = Variable definida por un script y que tan solo se usa en el ámbito de dicho script

En este ejemplo vamos a añadir la máquina *cliente1* como cliente de la plataforma, este cliente requiere de un módulo para sus scripts, ya que tiene scripts para la gestión de oracle que tan solo sirven en su entorno. Así pues el primer paso es crear el módulo en el servidor.
El contenido inicial del directorio */scripts* del cliente es:

```
cliente1:/scripts:#ls
MAX_USADOS         max_usados.sh      para_ORACLE.sh
levanta_ORACLE.sh  ocupa_discos.sh    recover_ORACLE.sh
```

4.1 Como crear un módulo

En primer lugar debemos crear una carpeta que se llame como el *hostname* del cliente, en este caso cliente1, y copiar en ella los archivos deseados:

cliente1:/scripts:#mkdir `hostname`

Acto seguido hay que copiar o mover a dicho directorio los scripts deseados, en este caso: levanta_ORACLE.sh, para_ORACLE.sh y recover_ORACLE.sh. Que son los encargados de gestionar Oracle.

cliente1:/scripts:# mv levanta_ORACLE.sh para_ORACLE.sh recover_ORACLE.sh `hostname`

El próximo paso es comprobar si los scripts deben incluir las cabeceras. En este caso no es necesario ya que se ejecutarán a mano y no desde el cron, así que *updater.sh* no tiene que realizar ninguna operación con él, así pues, los copiamos al servidor sin modificarlos:

cliente1:/scripts:#scp -r `hostname` server1:/scripts

Una vez copiado el directorio en el servidor, iniciamos sesión en el servidor, nos situamos en el directorio */scripts* del mismo y comprobamos que exista el directorio que acabamos de copiar, así como su contenido:

```
cliente1:/scripts:#ssh server1
root@server1:~# cd /scripts/
root@server1:/scripts# ls cliente1
levanta_ORACLE.sh  para_ORACLE.sh  recover_ORACLE.sh
```

Muy bien! todo está en su sitio, ya podemos ejecutar el último paso, crear el nuevo módulo en el servidor. Para ello ejecutamos el script del servidor *addmodule.sh* que se encuentra en el módulo del servidor (*server1*). Por preferencias personales, estos scripts no tienen permisos de ejecución, así que hay que ejecutarlos invocando a *sh*:

```
root@server1:/scripts# sh server1/addmodule.sh
Usage: server1/addmodule.sh "modulename to add"
```

Como se puede ver en el ejemplo, si se invoca un script sin pasarle los parámetros que necesita, este imprimirá el mensaje de uso, indicando entre comillas los parámetros que necesita. En este caso debemos indicarle el nombre del módulo que vamos a crear:

```
root@server1:/scripts# sh server1/addmodule.sh cliente1
```

Se creará el módulo, mostrando algo similar a esto (variarán los archivos pero no los mensajes):

```
U cliente1/levanta_ORACLE.sh
U cliente1/recover_ORACLE.sh
U cliente1/para_ORACLE.sh

No conflicts created by this import
cvs checkout: Updating cliente1
cvs add: cannot add special file `CVS'; skipping
cvs add: Re-adding file `levanta_ORACLE.sh' after dead revision 1.8.
cvs add: Re-adding file `para_ORACLE.sh' after dead revision 1.8.
cvs add: Re-adding file `recover_ORACLE.sh' after dead revision 1.8.
cvs add: use `cvs commit' to add these files permanently
cvs commit: Examining .
/scripts/cliente1/levanta_ORACLE.sh,v  <--  levanta_ORACLE.sh
new revision: 1.9; previous revision: 1.8
/scripts/cliente1/para_ORACLE.sh,v  <--  para_ORACLE.sh
new revision: 1.9; previous revision: 1.8
/scripts/cliente1/recover_ORACLE.sh,v  <--  recover_ORACLE.sh
new revision: 1.9; previous revision: 1.8
```

Con esto se acaba la creación de un nuevo módulo.

4.2 Adaptar los scripts de cron existentes, a la plataforma

En este paso adaptaremos los scripts existentes y que puedan ser de uso genérico a los requerimientos de la plataforma, en este caso: max_usados.sh, ocupa_discos.sh. Pero como ocupa_discos.sh ya existe en el módulo *generic* del servidor, tan solo adaptaremos max_usados.sh. Entonces abrimos *max_usados.sh* y en otra ventana abrimos */scripts/generic/updater.sh*, en el servidor, y copiamos las secciones necesarias desde *updater.sh* hacia *max_usados.sh*. la cabecera del archivo original es esta:

```
#!/usr/bin/sh

FILE=/tmp/FILE.$$
FILE_USADOS=/scripts/MAX_USADOS
VAR_TOTAL=/tmp/VAR_TOTAL.$$
```

```
V_MAX_USADOS=0
USADOS=0
...
```

En el original, tras la primera línea comienzan las acciones de la propia función del script, tras realizar los cambios necesarios el script queda así:

```
#!/bin/sh
        #### Global variables ###
if [ -f ../generic/global.conf ]; then
        . ../generic/global.conf
elif [ -f ./generic/global.conf ]; then
        . ./generic/global.conf
else
        echo "ERROR: generic/global.conf doesn't exists. Run
\"cvs:pserver:anonymous@server1:/scripts checkout
generic\" in the scripts folder to solve the problem."
        exit 1
fi

        #### Local variables ###
#Format del cron: minuts(0-59) hores(0-23) "dia del mes"(0-
31) mes(1-12) "dia de la setmana"(0-7, 0 i 7 son diumenge)
#For disable the automatic adition of this script into cron
comment this line
CRONDEF="00 8 * * 1,5 sh
$SCRIPTS/generic/max_usados.sh"

FILE=/tmp/FILE.$$
FILE_USADOS=/scripts/MAX_USADOS
VAR_TOTAL=/tmp/VAR_TOTAL.$$
V_MAX_USADOS=0
USADOS=0

### Include local variables, can overwrite the global ones
###
if [ -f $0.conf ]
        . $0.conf
fi
...
```

Hay que poner nuevo contenido tanto delante como detrás del contenido original, delante se incluye el archivo de configuración global (que incluye las variables necesarias para encontrar los scripts y para poder conectar al servidor), acto seguido se definen las denominadas variables locales: la orden que debe ejecutar el cron así como su periodicidad y las variables del script original, y finalmente (en caso de existir) se incluye el archivo de configuración del script, los archivos de configuración deben llamarse igual que el script añadiéndole la extensión *.conf*, en el caso que nos ocupa sería "*max_usados.sh.conf*" es muy importante no quitar la extensión *.sh* sino el script nunca incluirá el archivo de configuración.

Una vez realizados los cambios necesarios copiamos los scripts modificados al módulo *generic* del servidor y lo actualizamos desde el servidor:

cliente1:/scripts:#scp max_usados.sh server1:/scripts/generic

root@server1:/scripts# sh server1/updatemodule.sh generic

Con esto hemos acabado de adaptar los scripts al sistema.

4.3 Instalar el cliente

Para instalar el nuevo cliente, una vez realizados los puntos 4.1 y 4.2, hay que copiar el archivo */scripts/server1/installclient.sh* del servidor al cliente y ejecutarlo, este script comprueba que exista el directorio */scripts* y si no existe lo crea, acto seguido descarga los módulos *generic* y `*hostname*` y ejecuta el script *generic/updater.sh* que se encarga de realizar las modificaciones necesarias al crontab.

cliente1:/scripts:#scp server1:/scripts/server1/installclient.sh /

cliente1:/:#sh installclient.sh

cvs checkout: warning: failed to open //.cvspass for reading: No such file or directory
cvs checkout: Updating generic
U generic/control_fs.sh
U generic/default_action
U generic/global.conf
U generic/max_usados.sh
U generic/ocupa_discos.sh
U generic/updater.sh
cvs checkout: warning: failed to open //.cvspass for reading: No such file or directory
cvs checkout: Updating cliente1
U cliente1/crontab
U cliente1/default_action
U cliente1/levanta_ORACLE.sh
U cliente1/para_ORACLE.sh
U cliente1/recover_ORACLE.sh
cvs update: warning: failed to open //.cvspass for reading: No such file or directory
cvs update: Updating generic
cvs update: warning: failed to open //.cvspass for reading: No such file or directory
cvs update: Updating cliente1
…

Cada vez que se ejecuta el script *updater.sh* realiza un backup del *crontab* antes de modificarlo y lo deja en la *$HOME* del usuario que ejecute el script, en este caso es "/" y el archivo que ha generado es "crontab.backup.12_Nov_2007-15-41" hasta un máximo de 5, este parámetro se puede configurar en el propio archivo.
Ahora al ejecutar el comando *crontab –l* debería aparecer, como mínimo, una entrada que ejecute el script *updater.sh*, si no es así debemos asegurarnos que la línea *CRONDEF="00 1 * * * sh*

$SCRIPTS/generic/updater.sh" no esté comentada, si está comentada hay de descomentarla y ejecutar el script de nuevo.

4.4 Añadir, modificar y eliminar archivos de un módulo

El fin de esta plataforma es centralizar la gestión de los scripts, por lo tanto, lo más recomendable, una vez creado un módulo, es realizar cualquier acción (adición, modificación o eliminación) en el directorio del módulo en el servidor. A la hora de ejecutar los scripts de administración se tiene que hacer desde el directorio */scripts* o */scripts/módulo* esto es debido a que incluyen el archivo global.conf y buscan en la ruta *../generic/global.conf* o *./generic/global.conf*.

4.4.1 Añadir archivos

Ahora supongamos que creamos otra instancia de Oracle en el cliente *cliente1* para desarrollo, llamemosla *FIND* y creamos los scripts de control:

- levanata_FIND.sh
- para_FIND.sh

Una vez probados, los movemos al directorio */scripts/cliente1* del servidor (*server1*). Después iniciamos sesión en el servidor y ejecutamos el script *updatemodule.sh* que se encarga de añadir los nuevos archivos al CVS:

```
root@server1:/# cd /scripts
root@server1:/scripts# sh /scripts/server1/updatemodule.sh
cliente1
…
cvs commit: Examining .
/scripts/cliente1/levanta_FIND.sh,v  <--  levanta_FIND.sh
initial revision: 1.1
/scripts/cliente1/para_FIND.sh,v  <--  para_FIND.sh
initial revision: 1.1
…
```

Entre el resultado de la ejecución nos indica que ha añadido los archivos al módulo, para asegurarnos podemos ejecutar el script *listmodule.sh*:

```
root@server1:/scripts# sh server1/listmodule.sh cliente1
cvs rls: Listing module: `cliente1'
crontab
default_action
levanta_FIND.sh
levanta_ORACLE.sh
para_FIND.sh
para_ORACLE.sh
recover_ORACLE.sh
```

Y obtenemos el listado de los archivos contenidos en módulo, efectivamente aparecen los nuevos scripts, pero además aparecen los archivos "*crontab*" y "*default_action*" que no deberían estar aquí, así que vamos a eliminarlos mediante la ejecución del script *removefiles.sh*:

4.4.2 Eliminar archivos

```
root@server1:/scripts# sh server1/removefiles.sh cliente1
crontab default_action
…
/scripts/ cliente1/crontab,v  <--  crontab
new revision: delete; previous revision: 1.1.1.1
```

/scripts/cliente1/default_action,v <-- default_action
new revision: delete; previous revision: 1.1.1.1
There is a backup of files: crontab default_action in the folder /scripts/cliente1.backup_13-11-2007

Y al volver a listar el módulo, ya no aparecerán. Tal y como se ve en el mensaje, el script crea una copia de los archivos en una carpeta que se llama "*nobre-del-modulo.backup_dia-mes-año*".

4.4.3 Eliminar un módulo

Ahora se decide quitar la máquina *cliente1* del CPD, así que ya podemos eliminar el módulo para esta máquina, esto se realiza mediante el script removemodule.sh:

root@server1:/scripts# sh server1/removemodule.sh cliente1
…
The folder cliente1 is backed up to cliente1.backup_13-11-2007

Este script también realiza una cópia de seguridad de los archivos, en este caso de todo el directorio, y nos muestra, al final, donde ha realizado la cópia. Además de la copia que se realiza, los módulos eliminados continúan existiendo en el repositorio del servidor aunque marcados como eliminados, el directorio repositorio en *server1* es: */var/lib/cvsd/scripts/* y los scripts eliminados se almacenan en */var/lib/cvsd/scripts/nombre_del_modulo/Attic*. Para saber como recuperar versiones eliminadas, consultar documentación de CVS.

4.4.4 Rehacer un módulo eliminado localmente (en /scripts)

Si por cualquier motivo eliminamos un directorio de módulo completo (por ejemplo *cliente1*) y queremos recuperar la versión existente en CVS tan solo tenemos que ejecutar el script: updatemodule.sh

root@server1:/scripts# sh server1/updatemodule.sh cliente1

Este script volverá a generar el directorio y sus scripts.

4.4.5 Modificar archivos de un módulo

Para modificar archivos basta con editarlos en el directorios */script/modulo* del servidor (*server1*) y ejecutar el script *updatemodule.sh*, con esto quedarán reflejados los cambios en el servidor CVS y se replicarán a los clientes en la siguiente actualización.

4.5 Jugando con la configuración de los scripts

En este apartado veremos como personalizar la plataforma, tanto a nivel general como de una máquina en concreto, para ello imaginemos que queremos reducir la cantidad de backups del cron que realiza el script *updater.sh* de forma que todos los clientes guarden un máximo de 3 backups menos el cliente *cliente1*, que guardará 5 copias. Bien empecemos por realizar el cambio a nivel general.

1. Iniciar sesión en el servidor (*server1*)
2. Editar el script */scripts/generic/updater.sh*.
3. Sustituir *MAXBACKUPS=5* por *MAXBACKUPS=3* en el apartado de variables locales.
4. Ejecutar: **sh /scripts/server1/updatemodule.sh generic**
   …./scripts/generic/updater.sh,v <-- updater.sh
   new revision: 1.35; previous revision: 1.34

Ya está ahora tan solo hay que esperar a que los clientes actualicen y el cambio se aplicará a todos ellos. Pero ahora debemos asegurarnos, de que el cliente *cliente1* guarde 5 copias de seguridad, siguiendo estos pasos:

1. Iniciar sesión en *cliente1*
2. **cliente1:/:#cd /scripts/generic/**
3. **cliente1:/scripts/generic:#vi updater.sh.conf**
4. e introducimos la linea: *MAXBACKUPS=5*

Así el cliente *cliente1* guardará 5 copias de seguridad ya que el script *updater.sh* incluye el archivo *updater.sh.conf* después de la declaración *MAXBACKUPS=3* de modo que sobrescribirá la variable con la contenida en updater.sh.conf, es decir *MAXBACKUPS=5*.
Mediante estos pasos ya hemos realizado el cambio en todos los clientes, pero ¿Qué pasa si necesito que un cliente se actualice ya? Bueno pues tan solo se debe iniciar sesión en el cliente en cuestión y ejecutar el script */scripts/generic/updater.sh*, así los cambios se aplicarán inmediatamente.

Nota: Mediante un script de configuración local, podríamos cambiar la configuración de cron para un script en un cliente concreto.

5 Variables de configuración de la plataforma

En esta sección describiré las variables que usa la plataforma, su utilidad, así como en qué archivo se encuentran.

/scripts/generic/global.conf

SCRIPTS=/scripts

Esta variable especifica el directorio raíz de la plataforma, así que la raíz puede tener cualquier otro nombre, siempre y cuando se especifique en este archivo.

PSERVER=:pserver:anonymous@server1:/scripts

Mediante esta variable se especifica el servidor y login que se usará para las conexiones CVS. Es decir indica que máquina es el servidor y con que usuario y contraseña conectar.

/scripts/generic/updater.sh

MAXBACKUPS=5

Indica el número máximo de copias que se permitirán en cada cliente.

Variables en todos los archivos que se ejecuten mediante cron

CRONDEF="00 1 * * * sh $SCRIPTS/modulo/script.sh”

Esta es una de las variables más importantes de la plataforma, su contenido define la orden completa que se añadirá al cronttab, es decir que se añadirá todo el contenido encerrado entre comillas. Es muy importante encerrarla entre comillas ya que sino los asteriscos son interpretados por el shell y producen resultados inesperados. Justo encima de esta línea se encuentra un comentario que explica cada uno de los campos de cron. Otra recomendación es ejecutar los scripts mediante la ejecución de *sh*, de este modo: evitamos que no se pueda ejecutar un script por que no tiene permisos de ejecución y dejamos patente que lo que se ejecuta es un script y no un ejecutable.