Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Curso de Redes de Computadoras CI-4835
Trimestre Septiembre Diciembre 2013

Proyecto 2: Servidor de archivos en red con RMI

Objetivo del Proyecto

Que los alumnos comprendan y apliquen el esquema de programación cliente/servidor utilizando mecanismos de invocación remota de métodos.

Enunciado del Proyecto

Se desea implementar un sistema cliente-servidor de acceso a archivos remotos, basado en un servidor de archivos centralizado. El sistema manejará un esquema sencillo de validación de usuarios a través de un servidor de autenticación.
El cliente, una vez iniciado, podrá acceder a los servicios del sistema de archivo remoto mediante los siguientes comandos:

`rls`	muestra la lista de archivos disponibles en servidor centralizado.
`lls`	muestra la lista de archivos disponibles localmente (en el cliente).
`sub archivo`	sube un archivo al servidor remoto (Ej: `sub clase.pdf`). El archivo especificado como parámetro debe estar en la lista de archivos disponibles para el cliente localmente.
`baj archivo`	baja un archivo desde el servidor remoto (Ej: `baj ejemplo.c`). El archivo especificado debe estar en la lista de archivos disponibles en el servidor centralizado para que el comando funcione adecuadamente.
`bor archivo`	borra el archivo en el servidor remoto.
`info`	muestra la lista de comandos que el cliente puede usar con una breve descripción de cada uno de ellos.
`sal`	termina la ejecución del programa cliente, notificando este hecho a todos los procesos del sistema que lo requieran.

Cada archivo en el servidor centralizado tendrá asociado un propietario del archivo dentro del sistema (este es diferente al *owner* de *linux*). Todo archivo en el servidor está disponible para ser bajado, pero sólo el usuario propietario podrá borrarlo del servidor. Todo comando o acceso inválido arrojará un mensaje de error que el cliente podrá observar en su pantalla.
Los archivos que serán parte del sistema de archivos del cliente o del servidor de archivos serán aquellos que estén presentes en el mismo directorio donde se ejecute el programa.

Para propósitos de autenticación, el cliente inicia preguntando un nombre de usuario y clave. Los usuarios y claves podrán ser provistos en un archivo plano, uno en cada línea, con el formato siguiente:

```
nombre:clave
```

De no existir este archivo en el cliente, estos valores deberán ser leídos por pantalla.
Tanto los clientes, como el servidor de autenticación, reciben como parámetro el nombre del

archivo de usuarios. Todo comando enviado de un cliente al servidor de archivos debe contener al menos la tupla {nombre, clave}.
El servidor de archivos no podrá procesar ningún comando de un cliente que no esté autenticado, para ello el servidor de archivos cuenta con la ayuda de un servidor de autenticación que le permitirá verificar si un usuario realmente pertenece al sistema o no, usando para ellos los valores de nombre y clave del usuario.

El servidor de archivos, una vez iniciado, podrá acceder a una serie de servicios mediante los siguientes comandos:

`log`	Muestra una lista de los últimos 20 comandos que los clientes han enviado al servidor, identificando de que cliente provienen. En una misma línea debe estar la información del comando y el cliente que lo envió. No hace falta almacenar la respuesta del comando.
`sal`	Termina la ejecución del servidor de archivos.

La implementación deberá hacerse en Java, con invocación remota de métodos (RMI).

La sintaxis para la invocación del servidor de autenticación `a_rmifs` será la siguiente:

```
java a_rmifs -f usuarios -p puerto
```

Donde:
<`usuarios`> Es el nombre del archivo con los usuarios y claves
<`puerto`> Aunque normalmente en los procesos de invocación remota de métodos se pudiera usar un puerto por defecto, y no colocarlo aquí, por el hecho de tener varios proyectos de características similares en la misma instalación, solicitamos que los programas rmiregistry corran en un puerto que comience con el número 2 y que las siguientes 5 cifras correspondan a las últimas 5 cifras del carnet de alguno de los integrantes del grupo. Esta norma busca evitar que interactúen programas de proyectos diferentes.

La sintaxis para la invocación del servidor de archivos `s_rmifs` será la siguiente:

```
java s_rmifs -l puertolocal -h host -r puerto
```

Donde:
<`puertolocal`> Este será el puerto que usará el rmiregistry que tendrá información de los objetos remotos publicados por el servidor de archivos.
<`host`> Es el nombre DNS o dirección IP del computador donde corre el servidor de autenticación.
<`puerto`> Este será el puerto que usará el rmiregistry que tendrá información de los objetos remotos publicados por el servidor de autenticación.

La invocación del cliente `c_rmifs` se hace con el comando

```
java c_rmifs [-f usuarios] -m servidor -p puerto [-c comandos]
```

Donde:
<`usuarios`> Es el nombre del archivo con los usuarios y claves.
<`servidor`> Es el nombre DNS o dirección IP del computador donde corre el servidor de archivos.
<`puerto`> puerto donde estará el rmiregistry del servidor.
<`comandos>` será el nombre y dirección relativa o absoluta de un archivo, que contendrá en cada

línea, uno de los comandos que el cliente puede ejecutar por línea de comandos. El cliente debe ejecutar primero los comandos de este archivo y al terminar comenzar a aceptar comandos por teclado, a menos que uno de los comandos del archivo sea el comando `sal`.

Notas sobre la autenticación: El método usado en este proyecto se hace sólo con el fin de reforzar las habilidades de manejo de comunicaciones. Desde el punto de vista de la seguridad, no es un método válido a usar en cualquier instalación, puesto que cualquier atacante que capture un paquete tendrá la identificación de usuario y su clave, lo que constituye a una falla grave de seguridad.

ENTREGA

- La entrega debe hacerse en un archivo "tar.gz", siguiendo la convención del proyecto 1 (en este caso, "proy2grupoX.tar.gz") por el Aula Virtual, que incluya:
  - Código fuente
  - Archivo "LEEME.txt", que describa qué hace el programa, cómo se compila, cómo se ejecuta, qué requiere para ejecutarse, una breve descripción de su funcionamiento, limitaciones (si es el caso) y una explicación de cada uno de los archivos que componen el tar.gz
  - Debe incluir el archivo de compilación, de acuerdo a la herramienta utilizada (make, maven, etc.), debidamente comentado.

- El código debe estar debidamente documentado, siguiendo los estándares de documentación de Javadoc, y debe seguir las buenas prácticas de estilo de programación en Java.
- El tar.gz no debe contener nada más, por ejemplo, archivos compilados o temporales. Todos sus componentes deben estar identificados con nombre y carnet de los miembros del equipo y el número del equipo.

Condiciones de la entrega

La entrega se hará el martes de la semana 10. La hora precisa será especificada por el Aula Virtual.

Los equipos deben ser de dos estudiantes exactamente.

Los proyectos serán corregidos en los computadores del LDC, específicamente los de la sala Ernesto Leal; **si el proyecto no funciona en estos equipos se considerará que el proyecto no corre y no será corregido** por lo que, independientemente de donde elaboró el proyecto, debe probarlo en estos equipos con suficiente anticipación.

Información Adicional

- Al ser realizado el proyecto en equipo, cada uno de los miembros debe conocer plenamente todos y cada uno de los detalles de implementación del proyecto y podrá ser interrogado al respecto durante la corrección del mismo. Aquellas personas que no muestren un dominio de los detalles del proyecto no tendrán puntos en la evaluación.
- Cualquier caso de copia de proyectos será severamente castigado, no será evaluado ninguno de los proyectos involucrados y serán aplicadas las sanciones correspondientes establecidas en los reglamentos de la universidad.

GDRC