Asignatura: Laboratorio de Computadores. Curso 2007-08.
5º Semestre, 3er. Curso. Ingeniería Informática.

Práctica de SOCKETS

Especificación de la práctica:

"Un protocolo sencillo para transferencia de ficheros"

Área de Arquitectura y Tecnología de Computadores

Departamento de Informática e Ingeniería de Sistemas

En esta práctica vas a implementar un protocolo muy simple para transferir ficheros ASCII entre sistemas UNIX, siguiendo el modelo de comunicación *cliente-servidor* que ya conoces. Al protocolo lo llamaremos Simple File Transfer Protocol (sftp).

Utilizaremos como servidor un sistema SunOS que reside en la máquina **hendrix** y como cliente un sistema HP-UX que reside en la máquina **merlin**.

Simple File Transfer Protocol (SFTP).

SFTP es un protocolo para intercambiar ficheros entre dos máquinas UNIX utilizando datagramas UDP. El protocolo sólo permite transmitir ficheros desde la máquina donde reside el *servidor* hacia la máquina donde es ejecutado el *cliente*. Sólo los ficheros que se encuentran en el mismo directorio que el servidor pueden ser transferidos. Como ya sabéis, UDP transmite bloques de información de forma "no segura", de modo que vuestro objetivo será implementar SFTP añadiendo todo lo necesario para corregir errores.

Para simplificar el diseño, SFTP pone toda la inteligencia en el programa cliente. El cliente hará peticiones al servidor de trozos independientes de un fichero remoto y si no recibe el trozo después de cierto tiempo, deberá solicitarlo de nuevo. Te pedimos que implementes el programa cliente y como ayuda te proporcionamos el código fuente del servidor.

Especificaciones de SFTP.

SFTP es un protocolo de *petición-respuesta* "a pasos", es decir el cliente envía una petición al servidor de un trozo del fichero remoto que desea traerse y el servidor le envía dicho trozo, repitiendo el proceso hasta recibir el fichero completo.

A. El mensaje de petición del cliente contiene 6 campos en este orden:

1. Número de la versión (1-byte): contiene el número 1. Nos permitirá cambiar nuestro protocolo y seguir identificando paquetes de implementaciones más viejas.

2. Código de la operación (1-byte): contiene el código de la operación que realiza el cliente o el servidor. En esta versión de nuestro protocolo, la única operación que realiza el cliente es READ y se codifica con el número 1.

3. *Checksum* (2-bytes para un short integer en la ordenación de bytes de la red): contiene la suma complemento a uno de la petición (esta operación ya está implementada en la función xsum() que tienes en el código del servidor). Como UDP es "no seguro" y los bits pueden corromperse en el camino, deberás chequear todos los datos recibidos para asegurarte de que todo ha ido bien.

4. Identificador del fichero (4-bytes para un unsigned long integer en la ordenación de bytes de la red): como el paquete respuesta no contiene ningún campo con el nombre del fichero que se está transfiriendo, el servidor utilizará este identificador, que es único, para devolvérselo al cliente y así confirmarle que ese es el fichero que le pidió.

5. Desplazamiento (4-bytes para un unsigned long integer en la ordenación de bytes de la red): contiene el número de bytes que el servidor debe saltarse desde el comienzo del fichero antes de empezar a leer (lseek() a esta posición).

6. Nombre del fichero a transferir (64-bytes): contiene una cadena de caracteres acabados en NULL que corresponden al nombre del fichero. Esta cadena NO debe contener el carácter "/", es decir sólo podremos pedir ficheros que se encuentren en el mismo directorio donde se ejecuta el proceso servidor.

Timeouts: el cliente transmite una petición en un *datagrama* UDP al servidor y activa un temporizador de 1 segundo. Si el proceso cliente no recibe ningún mensaje válido después de pasado ese segundo, retransmite su petición activando ahora un temporizador de 4 segundos; si de nuevo durante este tiempo no recibe ningún mensaje válido, vuelve a retransmitir su petición activando un temporizador de 10 segundos; si tampoco se recibe un mensaje válido pasado este tiempo, el cliente nos informará de que la transferencia del fichero falló.

B. El servidor estará escuchando las peticiones de los clientes en un número de puerto conocido. El valor del puerto de escucha lo encontrareis en merlin en el fichero /users2/sistemas/sftp/port_sftpd ó en el fichero /users3/sistemas/sftp/port_sftpd (actualmente es el número 1024). Si el proceso servidor recibe una petición correcta, responderá con un mensaje que contiene 8 campos en este orden:

1. Número de la versión (1-byte): contiene el número 1.

2. Código de la operación (1-byte): contiene el código de la operación que realiza el cliente o el servidor. En esta versión de nuestro protocolo, la única operación que realiza el servidor es READ_REPLY y se codifica con el número 2.

3. *Checksum* (2-bytes para un short integer en la ordenación de bytes de la red): contiene la suma complemento a uno de la respuesta. El *checksum* debe ser correcto.

4. Identificador del fichero (4-bytes para un unsigned long integer en la ordenación de bytes de la red): coincide con el campo 4 de la petición del cliente.

5. Desplazamiento (4-bytes para un unsigned long integer en la ordenación de bytes de la red): coincide con el campo 5 de la petición del cliente.

6. Código de la respuesta (2-bytes para un unsigned short integer en la ordenación de bytes de la red): devuelve el estado de la operación. Un campo 0 indica que la petición fue válida y que la respuesta contiene el trozo del fichero pedido. Un campo mayor que 0x8000 indica que se detectó un error de protocolo en la petición. Un campo !=0 pero menor que 0x8000 indica que ha ocurrido un error en la lectura del fichero y el contenido del campo nos da el código de error en formato de <errno.h>.

7. Longitud (2-bytes para un unsigned short integer en la ordenación de bytes de la red): contiene el número de bytes de datos que hay en el campo *buffer*. Es el número de bytes del fichero dado a partir del campo desplazamiento y hasta un máximo de 512 bytes (este protocolo se basa de algún modo en lo que devuelve la llamada lseek(): no devuelve error si sobrepasamos el final del fichero y read() sólo devuelve "fin de fichero" -0 bytes leídos-, así que si con SFTP pedimos un trozo que pasa el final del fichero, recibiremos una respuesta válida pero con el campo longitud igual a 0).

8. *Buffer*: contiene los datos ASCII solicitados y no es mayor de 512 bytes.

El cliente será el responsable de confirmar la validez de todos los campos antes de aceptar los datos y escribirlos en su copia local del fichero.

El servidor sftpd permite ser lanzado con 3 parámetros:

-v	verbose flag	da información sobre los paquetes recibidos-enviados.
-r[frecuencia]	random flag	genera errores aleatorios.
-R[frecuencia]	less random flag	genera errores pero en una secuencia predefinida.

(Por ejemplo: el servidor de hendrix está lanzado con el parámetro: -r5 el cual hace que se estropeen 1 de cada 5 paquetes enviados).

Nota: La dirección IP de hendrix.cps.unizar.es es 155.210.152.16 y el servidor escucha en el puerto 1024. La dirección IP de merlin.cps.unizar.es es 155.210.4.33.

1. ¿Qué debes implementar?: 2 versiones del Simple File Transfer Protocol.

versión1) Utilizando lenguaje de programación C con llamadas al sistema, deberás implementar un programa *sftp cliente* que se invocará de esta forma:

```
sftpc [-v] get 1024@hendrix.cps.unizar.es fich_remoto [fich_local]
```

y que como resultado copiará el fichero_remoto del servidor (escuchando en el puerto 1024 de la máquina hendrix.cps.unizar.es) en el fichero_local de la máquina donde está el cliente (merlin.cps.unizar.es). Este cliente debe cumplir las especificaciones de SFTP que te acabamos de contar.

El parámetro **–v** es opcional y tiene el mismo significado que hemos dicho para el servidor:

-v verbose flag da información sobre los paquetes recibidos-enviados.

El nombre del fichero_local también es opcional y si no se pone, el fichero se guarda en la máquina donde está el cliente con el mismo nombre que el fichero_remoto.

Si falla la copia del fichero, tu programa deberá finalizar imprimiendo un mensaje de error.

versión2) De nuevo utilizando lenguaje de programación C con llamadas al sistema, deberás diseñar e implementar una versión mejorada de tu cliente sftpc y que no requiera cambios en el servidor. Un sftpc2 mejorado que consiga transferir ficheros en menos tiempo que la versión1.

2. ¿Qué dispones como ayuda?.

a. El código fuente del programa servidor tal y como está ejecutándose en hendrix y el fichero de *includes* (sftpd.c y sftp.h). Ambos ficheros los tienes en merlin en los directorios /users2/sistemas/sftp ó /users3/sistemas/sftp.

b. La documentación que viene a continuación: **"Programación en red sobre TCP/IP. Interface SOCKETS"**. En esta documentación encontrarás, paso a paso, toda la información que necesitas para implementar el protocolo SFTP que te pedimos. En ésta documentación también hay bibliografía recomendada.

c. El *manual del sistema* en merlin. Como ya sabes el manual de todo sistema operativo es la forma más directa de conocer cómo se invoca y funciona cualquier llamada al sistema. En cualquier caso y al final de esta documentación se ha incluido un extracto del *man* para aquellas llamadas relacionadas con el interface sockets y que deberás utilizar al implementar sftpc.

d. Consulta tus dudas con tu profesor. Para ello, acude a las Salas de Trabajo en el horario de reserva que tienes detallado más abajo ó acude al despacho del profesor durante su horario de tutorías.

3. Testea que sftpc y sftpc2 funcionan bien.

Junto al fuente del programa sftpd.c y en los directorios /users[23]/sistemas/sftp de merlin, dispones de los ficheros: test1, test2, test3 y mugande. Estos ficheros también estarán, en el mismo directorio que el servidor, en la máquina remota (hendrix).

Para comprobar que tus dos versiones del cliente funcionan correctamente, transfiere estos tres ficheros de la máquina remota a la local y compara sus contenidos con los originales del directorio /users[23]/sistemas/sftp. Para ello puedes usar el comando diff (ó cmp). Ejecuta diff (ó cmp) sobre el fichero transferido y sobre el original y deberás obtener la misma salida (mira en el *man* lo que hacen diff y cmp).

Reserva de las Salas de Trabajo:

Las salas de trabajo reservadas para el Laboratorio de Computadores son las siguientes:

- Todos los lunes (semanas A y B) de 10h a 14h en el laboratorio L0.04 (informática).
- Todos los lunes (semanas A y B) de 8h a 14h en el laboratorio L1.02 (redes).
- Todos los martes (semanas A y B) de 8h a 14h, y de 17h a 19h, en el laboratorio L1.02 (redes).
- Todos los miércoles (semanas A y B) de 19h a 21h en el laboratorio L0.04 (informática).
- Todos los jueves (semanas A y B) de 8h a 12h en el laboratorio L0.04 (informática).
- Todos los jueves (semanas A y B) de 10h a 12h, y de 17h a 19h, en el laboratorio L1.02 (redes).

Esta práctica de sockets puede trabajarse en ambos laboratorios, en cualquiera de los horarios descritos.

Es necesario que para esta práctica tengáis lo siguiente:

a. Esta documentación.

b. Cuenta de trabajo en *merlin*. Para ello **comprueba que recuerdas tu login+password**.

Si tienes algún problema con esto acude a la siguiente página Web: http://www.unizar.es/sicuz/siscen/index.html?menu=siscen y accede al enlace denominado "Solución automática de problemas de acceso al servidor de docencia". Rellenando y enviando el formulario que aparece en esta página, resolverás de forma rápida tu acceso a *merlin*.

c. Tu compañero de grupo. La práctica de Sockets se puede hacer en **grupos de 2 personas como máximo**. Al final de la Sesión de Presentación deberás decirle al profesor/a la composición de tu grupo.

Evaluación de la Práctica:

La práctica de Sockets (versión1 y versión2) será evaluada en 2 parciales y en un examen final.

1. La semana del **lunes 5 de Noviembre**.

Debe estar finalizada la versión1 de la práctica. Este primer parcial se evaluará demostrando el funcionamiento de la versión1 en *merlin*, en horario y laboratorio por determinar.

2. Desde el **28 de Noviembre hasta el 5 de Diciembre**.

Debe estar finalizada la versión2 de la práctica. Este segundo parcial se evaluará demostrando el funcionamiento de la versión2 en *merlin*, en horario y laboratorio por determinar.

3. La semana del **Lunes 7 de Enero**.

Este será el examen final de 1ª convocatoria, y las versiones 1 y 2 de la práctica serán evaluadas de la siguiente forma:

a) Deberéis entregar una documentación (ver anexos) donde se detalle cómo se han desarrollado cada uno de los clientes. Debéis contar cuáles han sido vuestras decisiones de diseño para las 2 versiones. Dicha documentación puede también incluir algún estudio de tiempos que muestre la mejora de la versión2 frente a la versión1.

b) Deberéis incluir en la anterior documentación un listado de los fuentes en C de los clientes de las 2 versiones.

c) Posteriormente a esta entrega, durante las semanas de exámenes, habrá una revisión individual del profesor con cada grupo acerca del trabajo realizado, con una demostración detallada en *merlin* del funcionamiento de los 2 clientes.

Anexo 1: Realización de la memoria [Actualización]

La realización de la memoria de la práctica tiene diseño libre, pero es recomendable que incluya los siguientes puntos:

1.- Resumen ejecutivo (2 paginas máximo)

2.- Problemas encontrados y posibles soluciones

3.- Justificación de la soluciones escogidas

4.- Explicación del desarrollo general del programa

5.- Pruebas de rendimiento

Anexo: Codigo fuente de los programas

Como única condición, el tamaño máximo debe ser de 15 páginas por una sola cara (sin contar el código fuente) y se recomienda un tamaño aproximado de unas 10 páginas.

Anexo 2: Forma de Entrega de la memoria [Actualización]

La forma de entrega de la memoria será por correo electronico de la siguiente manera:

- Un fichero ZIP con los siguientes ficheros:

    1. Memoria propiamente dicha en formato multiplataforma como PDF, OpenDocument .. (no en Word, por favor)

    2. Codigo fuente de la version 1 y version 2 en formato texto.

- Se debe mandar un único fichero por pareja

- El fichero se debe llamar labcom-p1_NIP-Alumno1_NIP-Alumno2.zip