Programación de sistemas distribuidos

Práctica 1: Objetos remotos

En esta primera práctica se suministran dos programas con la siguiente funcionalidad:

Programa 1: File Manager

Este programa sirve para listar los archivos de un directorio local, y realizar funciones básicas de lectura/escritura sobre ellos. Está compuesto por los siguientes archivos:

- Filemanager.cpp/h: Esta clase implementa las funciones de listado de un directorio local al ejecutable, pasado por parámetros al constructor. Tiene las siguientes funciones:
  - ListFiles : Devuelve un vector con los nombres de los ficheros encontrados en el directorio local. OJO, no es una lista dinámica, si se escriben nuevos ficheros no se añadirán automáticamente. Hay que liberar esa lista usando el método "freeListedFiles".
  - ReadFile: Si se le pasa un nombre de uno de los ficheros encontrados en el directorio, leerá su tamaños y contenido. Lo almacenará en las variables pasadas por parámetros.
  - WriteFile: Análoga al anterior, escribirá el contenido fichero en el directorio local. Si el fichero no existía en el directorio, se añadirá a la lista de ficheros.
  -
- Main_fm.cpp: Programa principal de prueba. Inicia una clase de tipo "fileManager", lista los archivos que hubiera en el directorio "dirprueba" (OJO, tenéis que crearlo y rellenar vosotros con vuestros archivos de prueba), e intenta escribir uno de los ficheros que se encontrara en un directorio local al ejecutable.

Programa 2: Multiplicación de matrices

Este programa implementa la multiplicación de matrices de NxM flas/columnas. Está compuesto por los siguientes archivos:

- MultMatrix.h/cpp: Esta clase implementta los métodos básicos de multiplicación de matrices. En concreto tiene los siguientes métodos:
  - Estructura matrix_t: Esta estructura es de uso interno, aunque podéis acceder a sus datos cuando sea necesario. Contiene los siguientes registros:
    - Rows y cols: Número de filas y columnas de las que está compuesta la matriz:

        - Data: Puntero a un array de números, con los datos de la matriz. En caso de matrices creadas correctamente, contendrá “rows * cols” números en formato integer de 32 bit.
    - ReadMatrix: Función de lectura de datos desde fichero. Se le debe pasar un nombre de fichero que contenga los datos de una matriz compatible con este programa. Leerá los datos y en caso de éxito devolverá una estructura matrix_t con los datos de la matriz.
    - WriteMatrix: Función análoga a readMatrix, escribirá una matriz almacenada en una estructura matrix_t en un fichero local al ejecutable.
    - MultMatrix: función que multiplica matrices. Recibe por parámetros dos estructuras matrix_t, y devuelve como resultado una tercera estructura matrix_t.
    - CreateIdentity/createRandom: Estas dos funciones sirven para crear matrices por defecto. La primera se rellenará con números aleatorios, la segunda sirve para crear una matriz identidad.
- Main_multMatrix.cpp: Programa principal para pruebas de esta clase. Crea una matriz random, una matriz identidad, las multiplica, el resultado es escrito a un archivo, se vuelve a leer, se vuelve a multiplicar, y se escribe por segunda vez el resultado.

Objetivos:

Se pide distribuir las clases fileManager y multMatrix, de tal manera que sea lo más transparente posible para los programas “main” suministrados. El alumno deberá implementar como mínimo dos programas por clase suministrada, pueden estar todos los programas en el mismo ordenador usando “localhost” y diferentes puertos:

- Programa 1 (servidor): Actuará como servidor, ofreciendo la implementación remota de las clases anteriores.
- Programa 2 (cliente):Actuará como cliente, sus programas harán las llamadas necesarias a las clases remotas contenidas en el servidor del programa 1.

OBLIGATORIO 5 puntos: Implementación básica de los programas distribuidos

Para aprobar la práctica, se pide implementar cuatro ejecutables, dos para cada programa suministrado. La función “main” de cada programa deberá modificarse lo mínimo para permitir su funcionamiento. Por cada programa suministrado, se pide lo siguiente:

- File manager: Deberá listar y realizar las operaciones de ficheros sobre un directorio en la máquina servidor. En el caso de realizar la operación “readFile”, se enviarán los datos desde el servidor al cliente. Y en caso de “writeFile” se enviarán los datos del fichero desde el cliente al servidor. Una de las acciones del ejecutable era la escritura de los datos leídos en un fichero local al ejecutable,

ese fichero deberá ser creado dentro de la máquina cliente (no está haciendo uso de métodos de la clase “fileManager”).

- Multiplicación de Matrices: Deberá realizar todos los métodos en funciones remotas. Los archivos leídos/escritos se crearán el servidor. Los datos se envían al cliente y de vuelta al servidor con cada uno de los métodos, según lo requieran.

+2 puntos: Mejoras en los programas MAIN de cada ejecutable.

Se podrán sumar hasta 2 puntos añadiendo funcionalidades a los programas main. Se ofrecen las siguientes funcionalidades:

- File Manager: Añadir una consola/menú que reciba comandos al estilo “linux shell”, y muestre al usuario. Los comandos son los siguientes:
  - Ls: Listar archivos del directorio remoto
  - Upload: Pedir un archivo local al usuario, y almacenarlo en el directorio remoto
  - Download: Pedir un archivo remoto, y almacenarlo en el directorio local

+2 puntos: Creación de un “broker de objetos”.

Se podrá sumar hasta 2 puntos creando un programa “broker de objetos”. Este programa estará ejecutándose en modo servicio en una máquina externa al cliente/servidor. Su objetivo será poner en contacto clientes con servidores de objetos. Mantendrá una lista de servidores activos, junto con las estadísticas de uso, para devolver al cliente el tipo de servidor más adecuado.

Su funcionamiento será el siguiente:

- Al iniciarse, quedará esperando conexiones de programas tipo cliente y servidor.
- Cada programa deberá identificarse al inicio de la conexión:
  - En caso de ser un servidor de objetos remotos, deberá indicar su IP y el nombre del tipo de objetos que sirve.
  - En caso de ser un cliente, deberá indicar el tipo de objetos que quiere instanciar.
- Una vez iniciada la conexión con alguno de los programas anteriores, deberá atender a las peticiones de la siguiente manera:
  - En el caso del servidor, almacenará sus datos (IP, y estadísticas de uso) en una tabla hash que usará como índice el nombre del tipo de clase que exporta.
  - En el caso del cliente, se buscará el nombre de la clase pedida en la tabla hash indicada anteriormente.
    - Si la encuentra, se devolverá al cliente la IP asociada
    - Si no lo encontrara, se quedará esperando a que algún server que exporte la clase pedida se conecte. Cuando eso ocurra, devolverá la IP asociada

Cambios en las clases “stub” e “imp” de cada programa

- Clase “stub” (cliente): Al ejecutar su constructor, se conectará al broker de objetos para pedir una IP de un servidor de un tipo de objeto. Una vez recibido, procederá a realizar la conexión con ese servidor
- Clase “imp” (servidor): Al iniciarse, se conectará al broker de objetos indicando el nombre de los objetos que exporta.

+1 punto: Calidad de código

Se valorarán los siguientes puntos para optar al 10 en esta práctica:

- Gestión correcta de memoria.
- Ejecución sin errores del programa
- Código “óptimo”:
    - Envío de mensajes mínimo
    - Algoritmos eficientes

EXTRAS:

La sección obligatoria (los primeros 5 puntos) no se puede modificar, pero se valorará cualquier añadido que el alumno desee implementar al margen de lo anterior. En esos casos, se puede sustituir la funcionalidad propuesta por el alumno por alguna de las mencionadas anteriormente. SE DEBE CONSULTAR CON EL PROFESOR ANTES DE REALIZAR ESOS CAMBIOS.

En cualquier caso, se da libertad de implementación, siempre que lo entregado por el alumno se corresponda con lo pedido en el enunciado. Ante cualquier duda de implementación PREGUNTAD ANTES AL PROFESOR.

Entrega

Las prácticas se pueden realizar por grupos de hasta dos personas. Ambos alumnos debe realizar la entrega del código desarrollado a través de blackboard, y realizar una defensa oral de la misma. Ambos recibirán la misma nota, y ambos deben estar preparados para responder cualquier pregunta sobre la práctica. En caso de que uno de ellos no sea capaz de responder, o sus errores en respuestas sean demasiado graves, se puede evaluar la práctica como suspensa para ambos. En resumen, los dos integrantes del grupo deben ser capaces de demostrar que han participado en el desarrollo.

Las prácticas se deben entregar antes del día **7 de Noviembre (23:59 del día 6)**, fecha en la que el profesor dará los horarios de defensa para los grupos que hayan entregado. En caso de querer presentarlas antes,

se podrá realizar la defensa en un horario concretado entre el profesor y los alumnos. Se deberá entregar lo siguiente:

- Fichero ZIP con los archivos generados/editados por los alumnos. Cada uno de los ficheros entregados DEBERÁ contener el nombre de los integrantes del grupo escrito en comentarios de código al inicio de cada archivo. Se deja a elección del alumno la estructura de directorios que contendrá su entrega.
- Nombre del fichero (OBLIGATORIO): PR1SISDIS_Alumno1_Apellido1_Alumno2_Apellido2.zip

Cualquier archivo entregado que no se adecúe a estas prescripciones podrá ser ignorado. Por favor, no entreguéis archivos con otros compresores o nomenclaturas.