Termostato Inteligente

Ejercicio N° 1

Objetivos	• Buenas prácticas en programación de Tipos de Datos Abstractos (TDAs) • Modularización de sistemas • Correcto uso de recursos (memoria dinámica y archivos) • Encapsulación y manejo de Sockets
Instancias de Entrega	**Entrega 1:** clase 4 (06/09/2016). **Entrega 2:** clase 6 (20/09/2016).
Temas de Repaso	• Uso de structs y typedef • Uso de macros y archivos de cabecera • Funciones para el manejo de Strings en C • Funciones para el manejo de Sockets
Criterios de Evaluación	• Criterios de ejercicios anteriores • Cumplimiento de la totalidad del enunciado del ejercicio • Ausencia de variables globales • Ausencia de funciones globales salvo los puntos de entrada al sistema (*main*) • Correcta encapsulación en TDAs y separación en archivos • Uso de interfaces para acceder a datos contenidos en TDAs • Empleo de memoria dinámica de forma ordenada y moderada • Acceso a información de archivos de forma ordenada y moderada

Índice

Introducción

El principal negocio de nuestra empresa son los termostatos para equipos de aire acondicionado hogareños y corporativos. Varias empresas se encuentran desarrollando sistemas inteligentes de control y nuestra compañía no se quiere quedar atrás.
Adoptar una estrategia de domótica [1] implica una inversión masiva y, antes de aprobarla, quieren realizar una prueba de concepto con funcionalidad mínima.
El sistema a desarrollar debe servir como *'hub'* (o concentrador) de temperaturas para permitir la generación de estadísticas con los distintos termostatos del edificio. A su vez, los termostatos requieren cierta inteligencia para informar estado al *hub*. Por último, el sistema de *hub* debe permitir el filtrado de la información recolectada, junto con estadísticas mínimas para ser mostrados en dashboards y alertas Web o de dispositivos móviles que la compañía está evaluando.

Descripción

El sistema a construir debe permitir su ejecución en dos modos: cliente y servidor.
Como cliente, se considera factible la ejecución dentro de ciertos microcontroladores [2] donde se debería capturar la temperatura del sensor embebido y transmitirla al *hub* mediante un simple protocolo de sockets.
Como servidor, se desea proveer el sistema concentrador que escuche la información reportada por cada sensor, haga ciertos cálculos estadísticos y almacene los resultados en archivos que serán consultados por procesos futuros para generar reportes con interfaz gráfica.
Tanto los clientes como el servidor contarán con acceso a la red local del edificio y podrán comunicarse libremente mediante TCP/IP. El sistema cliente será registrado en cada microcontrolador incluyendo la dirección IP asignada al servidor y serán los encargados de iniciar la comunicación.

A modo de simplificación de diseño, se espera que el servidor atienda una única conexión. Esto es, luego de atender a un cliente, la aplicación servidor finaliza.

El sistema cliente recolecta los valores de su sensor, corrige errores de medición y envía los datos al servidor en intervalos de 1 minuto y en modo ráfaga. Para hacer dicho envío se debe seguir el siguiente protocolo:

1. Establecer conexión con el servidor al iniciar el programa
2. Enviar el nombre del termostato seguido de '\n'
3. Esperar la aparición de valores sensados. Acumularlos hasta se detecte un cambio de minutos en el horario actual (ejemplo, pasó de 2016.08.16-20:51 a 2016.08.16-20:52)
4. Enviar el día y hora con formato usual (yyyy.MM.dd-hh:mm:ss). Cabe aclarar que el valor de 'ss' será siempre '00' ya que los cortes de intervalo se hacen en minutos exactos.
5. Enviar todos los valores de temperatura medidos desde el minuto anterior agregando un espacio (' ') antes de cada valor.
6. Enviar '\n' para indicar la finalización del listado correspondiente a la fecha dada.
7. Repetir esta operación mientras haya valores en el archivo indicado.
8. Cerrar la conexión cuando se detecta el cierre de archivo. Cerrar luego el programa.

El servidor, por su parte, almacena los valores recibidos y calcula un resumen diario con los siguientes datos:

```
<yyyy.MM.dd> <nombre-termostato> Max=<temperatura> Min=<temperatura>
Mediana=<temperatura> Muestras=<cantidad>
...
```

Esta información se imprime al detectar un cierre de la conexión del cliente y antes de finalizar el servidor. Se deben incluir todas las fechas que fueran recibidas por el servidor durante su conexión con el cliente en orden de aparición. Cabe aclarar que en el presente trabajo, **<nombre-termostato>** se encontrará repetido en el reporte, dadas las restricciones para que cada servidor atienda un único cliente.

Formato de Línea de Comandos

Para ejecutar en modo servidor, se llama al ejecutable indicando dicho modo junto con el puerto de escucha:

```
./tp server <port>
```

Donde **<port>** es un short que indica el el puerto donde escuchará el socket.

Para ejecutar en modo cliente:

```
./tp client <hostname> <port> <id-termostato> <millisec-time-step>
<date-time-now> <archivo-sensor>
```

Los parámetros **<hostname>** y **<port>** son el hostname/IP y el servicio/puerto del server remoto. **<id-termostato>** es el nombre identificador del termostato en el edificio. **<millisec-time-step>** indica el paso de tiempo entre cada muestra entregada por el sensor. Valores usuales de esta variable son 100 (10 muestras por segundo) o 500 (2 muestras por segundo) pero a fines de probar el funcionamiento del sistema se aceptarán valores de hasta 60000 (1 muestra por minuto). Esta frecuencia de muestreo es constante a lo largo de la ejecución del programa. **<date-time-now>** corresponde al día y hora de inicio de ejecución que permite, luego, calcular el tiempo de cada muestra teniendo en cuenta **<millisec-time-step>**. El formato de **<date-time-now>** es ‘yyyy.MM.dd-hh:mm:ss’ donde las siglas indican las magnitudes usuales.

Nota:
Se espera la encapsulación del objeto correspondiente a la fecha y hora. Con tal objetivo se recomienda el uso de funciones de parsing previstas por la biblioteca estándar tales como snprintf y sscanf. No se espera contemplar interfaces fuera de las necesarias para el presente trabajo. No es necesario implementar chequeo de año bisiesto.

Códigos de Retorno

El sistema debe retornar 0 (cero) si ejecutó exitosamente, independientemente de los valores de temperatura recibidos y el modo.
En caso de detectar error de conexión, debe retornar 1 (uno) sin realizar impresiones por consola.
En caso de poseer argumentos inválidos, debe retornar 2 (dos) sin realizar impresiones por consola

Entradas y Salidas

En modo servidor, el sistema no recibe información por entrada estándar. Utiliza la salida estándar para entregar información de log de cada cliente conectado y valores de termostato recibidos.

Por cada cliente conectado se debe imprimir la siguiente línea por salida de error:

```
Recibiendo termostato. ID=<nombre-termostato>
```

De la misma forma, por cada paquete recibido se debe imprimir la siguiente línea:

```
<yyyy.MM.dd-hh:mm:ss> - Datos recibidos: <cantidad de temperaturas recibidas>
```

Por último, por cada cliente desconectado se debe imprimir la siguiente línea:

```
Termostato desconectado. ID=<nombre-termostato>
```

En modo cliente, el sistema recibe los valores medidos por el sensor del termostato en formato binario y a partir de un archivo definido como argumento. Las mediciones aparecen de forma continua, con la frecuencia en la que fue configurado el sensor, utilizando dos bytes por valor medido en big-endian. Los sensores están calibrados de la siguiente forma:

- El valor mínimo medido (0x0000) corresponde a -17.0º C
- El valor máximo medido (0x02FF) corresponde a 59.7 ºC
- Los valores intermedios pueden calcularse con una escala lineal

Valores por fuera de estos rangos se consideran mediciones inválidas por lo que se debe repetir el valor de la anterior medición. En el caso particular de recibir una medición inválida como primer dato en la secuencia, se debe asumir el valor 0ºC. Por ejemplo:

```
A0 FF 01 5E 01 60 01 5F 55 55 01 64 01 64 01 60 ...
```

que corresponde a:

```
Error(0°C) 18.0°C 18.2°C 18.1°C Error(18.1°C) 18.5°C 18.5°C 18.2°C ...
```

En la salida de error se debe imprimir una línea por cada envío realizado desde el cliente al servidor. La línea debe incluir la fecha de envío en formato usual junto con todos los valores en formato leíble por el humano. Por ejemplo:

```
2016.08.22-23:59:00 - Enviando 10 muestras
```

Ejemplos de Ejecución

Iniciamos el servidor:

```
./tp server 9090
```

Iniciamos luego el cliente, asumiendo la IP 192.168.1.10 como destino y usando datos.dat como origen:

```
./tp client 192.168.1.10 9090 salon 1000 “2016.08.20-23:59:50” datos.dat
```

El cliente se conecta con el servidor y envía su nombre de termostato:

```
salon\n
```

Para indicar **datos.dat** se utiliza el siguiente contenido binario:

```
01 5E 01 60 01 5F 55 55 01 64 01 64 01 60 01 5E 01 60 01 5F 55 55 01 64
01 64 01 60
```

Luego, el sistema interpreta los siguientes valores:

```
18.0 18.2 18.1 18.1 18.6 18.6 18.2 18.0 18.2 18.1 18.1 18.6 18.6 18.2
```

Esto implica la siguiente transmisión de datos:

```
2016.08.20-23:59:00 18.0 18.2 18.1 18.1 18.6 18.6 18.2 18.0 18.2 18.1\n
2016.08.21-00:00:00 18.1 18.6 18.6 18.2\n
```

A su vez, el cliente imprime por salida de error:

```
2016.08.20-23:59:00 - Enviando 10 muestras
2016.08.21-00:00:00 - Enviando 4 muestras
```

Luego, se cierra la conexión. Notar que el inicio del sistema a las 23:59:50 define que para el minuto 23:59 se posean únicamente 10 segundos de recolección. Las 10 muestras en este caso coinciden con ello. El minuto siguiente, 00:00, no llega a completarse cuando se cierra la entrada de datos por lo que es enviado de forma parcial.

El servidor, por su parte, imprime información sobre la conexión y recepción en la salida de error:

```
Recibiendo termostato. ID=salon
2016-08-22-19:59:00 - Datos recibidos: 10
2016-08-22-20:00:00 - Datos recibidos: 4
Termostato desconectado. ID=salon
```

A su vez, recolecta los valores e imprime el sumario antes de finalizar incluyendo valor máximo, mínimo y mediana [3]. Dicho sumario es entregado por la salida estándar y tiene la siguiente forma:

```
2016.08.20 salon Max=18.6 Min=18.0 Mediana=18.1 Muestras=10
2016.08.21 salon Max=18.6 Min=18.1 Mediana=18.4 Muestras=4
```

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema debe desarrollarse en ISO C (C99).
2. Está prohibido el uso de variables globales.
3. Está prohibido el empleo de librerías y códigos de referencia para las listas ni la estructura de fecha/tiempo que se emplean en el sistema.

Referencias

[1] Domótica: https://es.wikipedia.org/wiki/Dom%C3%B3tica
[2] Microcontroladores: https://es.wikipedia.org/wiki/Microcontrolador
[3] Mediana: https://es.wikipedia.org/wiki/Mediana_(estad%C3%ADstica)