ADT_TCPSocket API
Interfaz de programación para conexiones TCP/IP en C++

Mario Chririnos Colunga
Áurea - Desarrollo Tecnológico

14 de septiembre de 2012

Índice

1. Introducción

En este documento se describe nuestra interfaz de programación de aplicaciones para conexiones TCP/IP utilizando **BSD sockets**. Se utiliza glib para monitorizar el estado del los puertos. El documento está estructurado de la siguiente manera: en la sección §2 describe el funcionamiento de nuestro código, la sección §3 describe detalles relacionados al ejemplo incluido con el código fuente y la sección §4 proporciona notas y comentarios finales sobre este documento.

2. ADT_TCPSocket API

2.1. Requisitos

Para compilar código de esta API se requiere contar con el paquete **libglib2.0-dev**, agregar al compilador C++ las banderas obtenidas mediante: `pkg-config --cflags glib-2.0`, `pkg-config --libs glib-2.0` y contar con un núcleo de Linux versión 2.4 o superior.

```
#ifndef ADT_TCPSOCKET_H
#define ADT_TCPSOCKET_H

#include <iostream>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <cstdlib>
#include <cstdio>
#include <cstring>
#include <glib.h>
#include <arpa/inet.h>

using namespace std;
//-----------------------------------------------------------------------------
#define CONNQUEUE        1 // connection backlog queue
#define BUFFERSIZE       255
//-----------------------------------------------------------------------------
class ADT_TCPSocket
{
 private:
        int rxid;
        GIOChannel* serverChannel;
        GIOChannel* clientChannel;

        int serverSocketFd;
        struct sockaddr_in server_addr;

        int clientSocketFd;
        struct sockaddr_in client_addr;

        int sendData(int fd, const unsigned char *data,  int length) const;
        int sendToServer(const unsigned char* data, unsigned int length) const;

        virtual void onGetData(struct sockaddr_in client);

        static int listen_callback(GIOChannel *source, GIOCondition condition, void *data);
        static int tcp_callback(GIOChannel *source, GIOCondition condition, void *data);

 protected:
        unsigned char* buffer;
        int bufferLength;

 public:
        int connectToServer(const char *_server, int portno);
        int sendData(const unsigned char* data, unsigned int length) const;
        int sendData(const char* data, unsigned int length) const;
        int sendData(const char *address, int port, const char *data,
int length) const;
        int sendData(const char *address, int port, const unsigned char *data,
int length) const;

        ADT_TCPSocket(int port);
        ~ADT_TCPSocket();
};
//-----------------------------------------------------------------------------
```

Figura 1: ADT_TCPSocket.h

2.2. Constructor

El constructor `ADT_TCPSocket(int port)` abre un puerto TCP/IP para que un cliente pueda estableser una conexión con el objeto creado.

2.3. Conexión

La función miembro `int connectToServer(const char *server, int portno)` permite conectarse el puerto *portno* en el servidor con dirección *server*. La función devuelve cero si la conexión fue exitosa.

2.4. Envío de datos

Para enviar datos a través de la conexión TCP/IP están disponibles dos funciones:

- `int sendData(const unsigned char* data, unsigned int length)`
  La función envía un arreglo de datos *data* de longitud *length*. Esta función debe ser utilizada cuando exista una conexión entre un un cliente y un servidor; Cuando el programa actúa como cliente la conexión con el servidor es creada por medio de la función `connectToServer(.)`; Cuando el programa es usado como servidor los datos son enviados a la conexión creada con el cliente.

- `int sendData(const char *address, int port, const char *data, int length)`
  Esta función crea una conexión en el puerto *port* con el servidor *address* y envía un arreglo de datos *data* de longitud *length*. Al terminar de enviar los datos la conexión se cerrada.

2.5. Eventos

La API utiliza glib para monitorizar el estado del puerto TCP/IP y llamar a una función de retro llamada (*callback*) cuando se genere un evento. Se debe iniciar un *g main loop o gtk main loop* en el programa principal para activar los eventos.

La función virtual miembro `virtual void onGetData(struct sockaddr_in client)` es llamada cada vez que se reciben datos por el puerto TCP/IP, esta función puede ser sobreescrita por clases derivadas.

3. Ejemplo

Junto con el código de esta API se provee de un programa ejemplo para demostrar su funcionamiento.

3.1. Requisitos

Para compilar el programa ejemplo se requiere contar con el paquete **libgtk2.0-dev**. Las banderas de compilación se especifican en el archivo *makefile* de este ejemplo.

4. Notas

El código fuente de esta API puede ser descargado en nuestro sitio web, en donde también se pueden reportar errores en el código fuente. Para reportar errores en este documento favor de escribir a errata@aurea-dt.com.