

```

//Editor-Info: *- C++ *-
//
//Subject: Calypso-IP
//
//File: routeadapter.h
//
//Version: $Revision: 1.2 $
//
//State: $State: Exp $
//
//Date: $Date: 2000/07/31 08:58:57 $
//
//Organisation:
// Helsinki University of Technology
// Laboratory of Telecommunications Software and Multimedia
//
//Author:
// Juha Hyttinen
//
//Description:
// Example how to use pfNETLINKsocket.
//
//Copyright:
// Copyright 1999 Helsinki University of Technology
// ALL RIGHTS RESERVED BETWEEN JUNE 1999 AND JANUARY 2002.
//
//Licence:
//
//
//History:
//
//

#ifdef __linux__
#ifdef __ROUTEADAPTER_H__
#define __ROUTEADAPTER_H__

#include "pf/netlinksocket.h"

#include <list.h>
#include <algo.h>

class routeAdapter : public pfNETLINKsocket
{
    public:
        // hee, this example is singleton
        static routeAdapter *instance();

        // This is for first time calling. Its initializes buffers.
        static routeAdapter *instance(pfUlong pduSize_);

        // This might return pfConduit if is needed

```

```

// (If adapter is added to conduit stack!)
void createNETLINKConnection(int netlinkFamily_);

routeAdapter(pfUlong pduSize_);
virtual ~routeAdapter(void);

void readAction(pfFrame &frame_, pfUlong code_);
void writeAction(pfUlong code_);

// If we want send one request at time. If there is information in
// buffer, that will be also sended (Can't just discard)
pfBoolean addRoute(int family_, pfUlong dst_, pfUlong mask_,
    pfUlong gw_, pfUlong metric_, pfUlong priority_,
    pfUlong oif_, pfByte scope_ = RT_SCOPE_UNIVERSE);

pfBoolean delRoute(int family_, pfUlong dst_, pfUlong mask_,
    pfUlong gw_, pfUlong metric_, pfUlong priority_,
    pfUlong oif_, pfByte scope_ = RT_SCOPE_UNIVERSE);

// If we want send many requests at time, we put them to buffer
void addRouteBuffer(int family_, pfUlong dst_, pfUlong mask_,
    pfUlong gw_, pfUlong metric_, pfUlong priority_,
    pfUlong oif_, pfByte scope_ = RT_SCOPE_UNIVERSE);

void delRouteBuffer(int family_, pfUlong dst_, pfUlong mask_,
    pfUlong gw_, pfUlong metric_, pfUlong priority_,
    pfUlong oif_, pfByte scope_ = RT_SCOPE_UNIVERSE);

void getRouteTable(int family_);
void readInterfaces(void);

// This commits all requests from buffer
pfBoolean commitBuffer(void);

void clearReadRequest(void);

void setLockFlag(pfBoolean value_);
pfBoolean getLockFlag(void);

protected:
private:

// get next free sequence number
pfUlong getNextSeq(void);

// These parses pfFrame for each request
pfFrame parseFrame(int family_, int frametype_, pfUlong dst_,
    pfUlong mask_, pfUlong gw_, pfUlong metric_,
    pfUlong priority_, pfUlong oif_, pfByte scope_);

// Buffer for requests
pfFrame _requestBuffer;
pfBoolean _requestBufferLock;

// next free sequency number.

```

```
    pfUlong _seq;  
    pfBoolean _lockFlag;  
  
    static routeAdapter * _only;  
};  
#endif // __ROUTEADAPTER_H__  
#endif // __linux__
```