PHDDNS Embed开发说明

目录

[PHDDNS Embed开发说明 1](#_Toc243910279)

[1概述 1](#_Toc243910280)

[2源码清单 1](#_Toc243910281)

[Windows编译 1](#_Toc243910282)

[Linux编译 2](#_Toc243910283)

[C++版本Embed/src清单 2](#_Toc243910284)

[纯C版本Embed/src/c清单 3](#_Toc243910285)

[3 C++开发步骤 3](#_Toc243910286)

[4 纯C开发步骤 5](#_Toc243910287)

# 1概述

本套源码用于嵌入式开发商快速进行花生壳DDNS的编译及嵌入，分为C++/C两个版本，包含已经写好的开发示例以及相关依赖；

说明：

* 本套源码在以下平台编译通过：
  + Windows: VC6.0
  + Linux : gcc4.1.2

./configure

* + Linux: arm-linux-gcc3.4.1

./configure --host=i386-linux --target=arm-linux --build=arm-linux CXX=arm-linux-c++ CC=arm-linux-gcc LD=arm-linux-ld

* 客户端实现内部不会开启线程
* 对于异常断线的重连问题，代码内部已作处理

# 2源码清单

## Windows编译

以下源码都包含VC6工程文件，可在Windows上编译测试，无其他依赖项

命令行：

**embed.exe <服务器地址> <用户名> <密码>**

## Linux编译

1. 解压缩

#tar zxvf phddns-22.0.tar.gz

# cd phddns-22.0

1. 编译前configure

#./configure 为当前平台编译

#./configure --host=i386-linux --target=arm-linux --build=arm-linux CXX=arm-linux-c++ CC=arm-linux-gcc LD=arm-linux-ld 使用arm-linux工具链交叉编译到arm平台，相关工具链的安装请参考开发板文档

1. 编译最终可执行文件

#make

说明：该过程如果执行成功，将自动生成两个可执行文件：Embed/src/phddns以及Embed/src/c/phddns，分别为C++版本及C版本，功能完全相同，参数与Windows版本一致。

## C++版本Embed/src清单

|  |  |
| --- | --- |
| **main.cpp** | **开发示例** |
| Makefile.am | Linux编译Makefile |
| PHGlobal.cpp  PHGlobal.h | 全局变量及公用函数实现 |
| phupdate.cpp  phupdate.h | PHDDNS Embed网络通信主过程 |
| common/base64.cpp  common/bitstream.cpp  common/bitstream.h  common/blowfish.cpp  common/blowfish.h  common/coder.cpp  common/coder.h  common/global.h  common/lutil.h  common/md5.cpp  common/md5.h | 通用加密算法相关代码以及数据类型定义(global.h)，其中包括：base64、md5、blowfish |
| common/generate.cpp  common/generate.h | DDNS嵌入式私有专用加密算法实现，用于TCP主认证过程，以及每个心跳包的加解密 |
| common/log.cpp  common/log.h | 日志实现 |
| common/PHSocket.cpp  common/PHSocket.h | SOCKET封装，支持TCP/UDP |

## 纯C版本Embed/src/c清单

|  |  |
| --- | --- |
| base64.c  blowfish.c  blowfish.h  lutil.h  md5.c  md5.h | 通用加密算法相关代码，其中包括：base64、md5、blowfish |
| phsocket.c  phsocket.h | Socket C封装 |
| phglobal.c  phglobal.h | 全局控制结构申明 |
| phupdate.c  phupdate.h | PHDDNS Embed网络通信主过程 |
| generate.c  generate.h | DDNS嵌入式私有专用加密算法实现，用于TCP主认证过程，以及每个心跳包的加解密 |
| log.c  log.h | 日志实现 |
| **main.c** | **开发示例** |

# 3 C++开发步骤

注意：

* 下文中的代码仅作为示例，相关函数和参数的详细说明请参照doc/html/index.html）
* Windows平台推荐使用COM组件方式进行开发，详见《PhEmbed Windows平台COM组件.docx》

1. 从Oray相关部门取得以下四个厂商专用信息（每个厂商都不同），用于Oray监控各厂商设备的连接情况及防止滥用：

* 嵌入式ID --2个字节
* 嵌入式版本号 --2个字节
* 嵌入式认证码 --4个字节
* 专用服务器地址 --如：phembed\_sample.oray.net

1. 组合嵌入式ID与嵌入式版本号到4个字节的整数

如0xFFFF为嵌入式ID，0xEEEE为嵌入式版本号，则组合后为：0xFFFFEEEE

1. 重载CUpdateBase类用于取得通信过程中的消息和数据，如：

class CPhUpdater : public CUpdateBase

{

public:

protected:

virtual void OnStatusChanged(int status, long data)

{

printf("OnStatusChanged %s\n", onvert\_status\_code(status));

if (status == okKeepAliveRecved)

{

printf("IP: %d\n", data);

}

if (status == okDomainsRegistered)

{

printf("UserType: %d\n", data);

}

}

virtual void OnDomainRegistered(std::string domain)

{

printf("OnDomainRegistered %s\n", domain.c\_str());

}

virtual void OnUserInfo(std::string userInfo)

{

printf("OnUserInfo %s\n", userInfo.c\_str());

}

virtual void OnAccountDomainInfo(std::string domainInfo)

{

printf("OnAccountDomainInfo %s\n", domainInfo.c\_str());

}

private:

};

1. 示例化重载后的CPhUpdater，并配置相关参数：

CPhUpdater phup;

phup.phglobal.clientinfo = 0xFFFFEEEE; //这里填写刚才第二步算出的值

phup.phglobal.challengekey = 0x12345678; //这里填写**嵌入式认证码**

strcpy(phup.phglobal.szHost,

"phembed\_sample.oray.net"); //你所拿到的服务器地址

strcpy(phup.phglobal.szUserID,

"username"); //Oray账号

strcpy(phup.phglobal.szUserPWD,

"password"); //对应的密码

1. 进入花生壳DDNS更新循环

#ifdef WIN32

#define sleep(x) Sleep(x\*1000)

#endif

while (!bStop) //bStop用于控制是否退出

{

int next = phup.step();

sleep(next); //next为秒数

}

phup.stop();

1. 此时重载类实例虚函数将获得事件与数据回调。

# 4 纯C开发步骤

请参考C++开发步骤与Embed/src/c/main.c，所有参数名都一致，仅是状态跟数据采用了回调函数的形式。

Oray研发中心

2009/10/21