thunder09的专栏--没事就写点儿

Life is a journey.

- 博客首页
-
- 全站
- 搜索

- 空间
- 博客
- 好友
- 相册
- 留言

用户操作

[留言] [发消息] [加为好友]

订阅我的博客

1 位读者

POWERED BY FEEDSKY

thunder09的公告

thunder09的后花园.

文章分类

- IBM SOA /WebSpere /ESB
- interview
- Java 开发和发布
- microsoft SOA
- WinForm开发
- 程序开发
- 内容管理系统CMS
- 设计、软件工程与项目管理
- 数据库
- 有趣有趣
- 中间件

存档

- 2010年11月(14)
- 2010年10月(3)
- 2010年09月(1)
- 2010年08月(6)
- 2010年06月(2)
- 2010年05月(18)
- 2010年04月(6)
- 2010年03月(33)
- 2010年02月(13)

用NET-SNMP软件包开发简单客户端代理

收藏

用NET-SNMP软件包开发简单客户端代理

写在前面的话：

对于net-snmp我也是一个初学者，开始学习时也碰到了很多低级的问题。在很多论坛上（事实上比较少^_^, 建议大家直接去sourcefoge社区看关于net-snmp的mail-list），都没有比较初级入门的文章，本着开源学习的精神，把自己的一点收获，共享给大家。通过参考一些前辈的文章和帮助文档，本文实现了一个简单的mib，并编写了文档。本文主要面向初级学习者（我也是个菜鸟），欢迎大家留言讨论。

作者：solomoon

完成时间：2005-9-11

Email：lilofreeman@yahoo.com.cn

Web：http://bibu.blogchina.com

用NET-SNMP软件包开发简单客户端代理

1 SNMP协议简介

作为一个完备的系统，必须有一套反馈机制来调整系统的运行。简单网络管理协议产生的目的，就是为了使松散的网络更加有效地运行。它广泛的应用于监测网络的状态、网络设备的运行情况、各种电脑设备以及一些辅助的外围设备，使得网络管理员通过对节点的查询和设置，发现并定位故障，进而采取相应措施维护网络。网络管理的研究已经发展了许多年，对于日益纷繁的需求，简捷性和扩展性仍是研究的主题。本文档的目的是关于客户端代理的开发，不是对协议发展的探讨。本文中协议相关资料可以参考RFC文档：

RFC1155：Structure and Identification of Management Information for TCP/IP-based Internets

RFC1157： SNMP

RFC1212： Concise MIB Definitions

RFC1215： A Convention for Defining Traps

RFC1905： Protocol Operations for SNMPv2

RFC2011： SNMPv2 Management Information Base for the Internet Protocol using SMIv2

RFC2578： Structure of Management Information

RFC2579： Textual Conventions

RFC2580： Conformance Statements

1.1 网络管理协议结构

SNMP的网络管理模型包括以下关键元素：管理端、代理端、管理信息库、网络管理协议。它基于tcp/ip协议，属于应用层协议，通过udp协议通信。管理端与代理端的通信原语包括：Get,Getnext,Set,Trap。对应这些命令相应的SNMP结构框架实现如图1所示

图1. SNMP实现结构图

从上图我们可以看到协议，消息传递方式等。另外，在udp数据包中，发送信息是按ASN.1自解释方式编码的。但对于许多小型被监管设备，可能会运行不同协议，或者运行完整代理花费很大，于是产生了代管设备，主代理和子代理的概念。在小型设备上运行子代理，把数据发给主代理来完成snmp协议的通信。

1.2 管理信息库

SNMP以MIB（管理信息结构）为基础来描述被监管资源，由此建立的数据集和称之为MIB

库。它是一种树型结构的数据库，被监管的对象都处于叶子节点上。每个被监管对象都由一个唯一的对象标识符来识别。对象信息的存储结构由MIB定义的简单变量和表来构造，它一般包含描述名（对象标识符）、数据类型、读写规则、功能描述、状态。MIB的定义

可以查询RFC1155，它定义了四种基本数据类型：INTEGER，OCTET STRING，OBJECT IDENTIFIER和NULL。由这四种基本类型通过SEQUENCE构造列和表，以及新类型如：NetworkAddress、IpAddress、Counter、Gauge、TimeTicks、Opaque等，以及宏定义。当然，根据需要还可以构造自己的数据类型。

1.3 SNMP的版本

目前SNMP有三个版本snmpV1、snmpV2、snmpV3。针对原始的V1版，93版的v2加入了安全机制，但用户对其并不感兴趣，在96版的v2中又删除了安全机制，99年开始酝酿的v3版开始提出一个snmp的统一架构，采用User-based安全模型和View-based访问控制模型提供SNMP网络管理的安全性。安全机制是SNMPv3的最具特色的内容。

2 SNMP开发软件包

目前，开发SNMP的软件包有许多可以选择如SNMP++、AGENT++、NET-SNMP等。这里我们选用的是NET-SNMP。首先它是一个开源软件，其次基于C语言开发，便于移植。ucd-snmp源自于卡耐基.梅隆大学的SNMP软件包CMU snmp 2.1.2.1, 由加州大学Davis分校(University of California at Davis)开发与维护, 所以命名为ucd-snmp。2000年11月ucd-snmp项目转到由SourceForge(www.sourceforge.net)管理, 并更名为net-snmp。

2.1 NET-SNMP简介和安装

net-snmp早先是在Unix平台下开发的。现可以移植到：

* HP-UX (10.20 to 9.01 and 11.0)
* Ultrix (4.5 to 4.2)
* Solaris SPARC/ULTRA (2.8 to 2.3), Intel (2.9) and SunOS (4.1.4 to 4.1.2)
* OSF (4.0, 3.2)
* NetBSD (1.5alpha to 1.0)
* FreeBSD (4.1 to 2.2)
* BSDi (4.0.1 to 2.1)
* Linux (kernels 2.4 to 1.3)
* AIX (4.1.5, 3.2.5)
* OpenBSD (2.8, 2.6)
* Irix (6.5 to 5.1)
* OS X (10.1.1 and 10.1.2)
* Dynix/PTX 4.4
* QNX 6.2.1A
* Windows

等多个平台。Net-snmp是一个代理端软件，但也提供管理端的查询工具。安装有两种方式：一是直接安装的二进制包，二是需要编译的源代码。我们在windows平台上安装的二进制包，在虚拟Unix平台CygWin上编译安装的源代码。在CygWin中，按照常规的configure, make ,make install三个步骤就可成功编译安装源代码。在windows上的二进制包的安装就非常简单了，只需按提示就可完成。源代码和二进制包可从http://www.net-snmp.org/网站下载，本文中所用的是net-snmp5.2.1.2的版本。之所以要先安装一个可运行的net-snmp系统，是因为我们开发程序运行环境的配置文件，是按照默认安装路径内部设定搜索的；另外，还可以利用其提供的配置工具来生成配置文件，利用提供的查询工具来测试程序。

2.2 NET-SNMP代理的配置

运行net-snmp之前先要进行环境设置，否则无法查询到结果。环境配置文件由snmpconf命令交互生成。运行snmpconf后，提示有三个配置文件：snmpd.conf，snmptraps.conf，snmp.conf。其中，snmpd.conf用来配置代理和管理端通信时的参数，只需设置两个参数就可正常运行程序了，一是community name，有只读rocommunity和读写rwcommunity之分，相当于访问账号，这里设rocommunity为public；另一个是访问端口，设为snmp协议默认的161端口。 Snmp.conf是与mib库设置相关的配置文件。Snmptraps.conf用来设置代理陷阱，本文没有讨论陷阱。配置文件可以放在三个地方，一是盘符根目录下，二是~\usr\etc\snmp目录下，三是~\usr\snmp\persist，按标准路径最好是第二种方式。

另外，snmpconf和mib2c工具都是基于perl脚本的，在windows下需要安装perl才能运行。按照帮助文档的提示，下载ActivePerl安装。并按照帮助文档中perl的安装要求，下载在win32环境下所需的其他组件，配置并测试perl模块，使snmpconf和mib2c能正常运行。

2.3 NET-SNMP工具的使用

当环境设置好后，运行snmpd.exe，即snmp代理进程，就可以使用管理工具查询其中的信息了。Net-snmp提供的查询工具有很多，这里只介绍常用的几个，而且大部分查询命令的格式都大同小异。这里以.iso.org.dod.internet.mgmt.mib-2.system为例，其Oid为：.1.3.6.1.2.1.1。结构如下：

……system .1.3.6.1.2.1.1

|——sysDescr .1.3.6.1.2.1.1.1

|——sysObjectID .1.3.6.1.2.1.1.2

……

1) snmpget.exe——snmpget [OPTIONS] AGENT OID [OID]...用来查询叶子节点

实例：snmpget –v2c –c public localhost .1.3.6.1.2.1.1.5.0

-v2c： 使用的是2c的snmp版本，可选1|2c|3

-c public：community 名为public

localhost: 代理的地址，这里因为代理运行在本机上，所以可用localhost

这里查询的是.iso.org.dod.internet.mgmt.mib-2.system.sysName，其Oid为.1.3.6.1.2.1.1.5，使用这个命令使叶子节点要在后面加.0。

2） snmpgetnext.exe——snmpgetnext [OPTIONS] AGENT OID [OID]...通过父节点查询叶子节点

实例：snmpgetnext –v2c –c public localhost .1.3.6.1.2.1.1

这个命令假设不知道叶子节点，但知道父节点，则可遍历到第一个叶子节点。此例结果等同于上一个例子。Oid也可输入.1.3.6.1.2，因为它是按字典顺序遍历的。

3） snmptable.exe——snmptable [OPTIONS] AGENT TABLE-OID 用来查询表对象

实例：snmptable –v2c –c public localhost .1.3.6.1.2.1.4.20

这个命令查询表对象，本例中查询的是.iso.org.dod.internet.mgmt.mib-2.ip.ipAddrTable

4）snmpset.exe——snmpset [OPTIONS] AGENT OID TYPE VALUE [OID TYPE VALUE]... 修改数据

实例：snmpset –v2c –c public localhost .1.3.6.1.2.1.4.21.1.3.x i 99

x：在这里是索引值，表示表项中某一列的第几个数据,根据要求设定

i: 这里是列数据类型，包括i: INTEGER, u: unsigned INTEGER, t: TIMETICKS,

a: IPADDRESS o: OBJID, s: STRING, x: HEX STRING,

d: DECIMAL STRING, b: BITS U: unsigned int64,

I: signed int64, F: float, D: double

5） mib2c 用来把mib库文件编译成.c和.h模版。具体使用在下面章节的应用中介绍

3 扩展开发——代理

当系统加入了新设备，或设备配置发生了变化等，需要实现新的mib模块，这时就需要扩展代理端了。在利用net-snmp做扩展之前的准备工作是，一下载源代码net-snmp5.2.1.2，二编译出库文件：netsnmpagent.lib、netsnmphelpers.lib、netsnmpmibs.lib、netsnmp.lib。这四个库的工程文件都位于~\net-snmp-5.2.1.2\win32\下对应的文件夹中。编译库文件时注意把netsnmp.lib放到最后编译，它可能会参考一下前面的编译文件。对于其他进一步的应用和开发请参考帮助文档编译和使用其他的工具；本文档中实现的代理程序，在源代码包中只需要引用这四个库文件。

开发工具VC6.0的环境设置也需要注意，参考帮助文档readme.win32，设置好include目录，library目录。在project\configure的选项link的相关栏目中添加上四个库文件，还要加上wsock32.lib库。另外，在程序编译过程中会碰到与VC的默认库相冲突的地方，按提示在上述添加库的地方加上/NODEFAULTLIB:XXX.LIB来除去默认库。

代理的开发过程基本上遵循：mib模块→转换成C文件→编译进代理中。

3.1 NET-SNMP中的scalar对象和table对象

mib模块一般都由变量和表组成。因此Net-snmp把SMI中的对象分为两大类：scalar和table。Scalar就包含我们常用的整型，字符串，时间等等数据类型。table就是scalar的一种集合，有一个和多个列组成，类似于数据库中的表。它必须具有索引项，用来按一定顺序检索表项。

Mib2工具通过模版把mib文件解析成.c和.h文件，这些文件仅仅是半成品，还需要手工在相应地方添加相应代码。Mib2c有很多模版，可以根据相应需要来调用不同的模版。但mib2c

目前不支持同时解析scalar和table 对象，对于具有这两种对象的mib模块，需要分别生成代码文件，然后再合并成整体。

3.2 NET-SNMP扩展代理的两种方式

用net-snmp扩展代理，实现方式可归结为两种：一是静态库方式，通过修改配置头文件，在相应地方包含新引入的mib模块的.c和.h文件，然后重新编译库文件和代理程序；二是编译动态共享库，只需把新引入的mib模块的.c和.h文件编译成动态库，通过设置能够让代理程序载入。

对于第二种方式，一需要编译成.so动态共享库，二需要原代理程序是否包含dlmod或load命令，三还要看系统是否支持。一般情况下仅支持Unix平台。我们在CygWin下试验过Unix平台编译。在VC下的编译环境，基本上都是参考其makefile,configure等文件。

因为是在windows平台下开发，本文档采用的是第一种方式。这种方式允许我们在原有mib库上添加新的mib模块，也可以只针对需要的mib模块编译单独的程序。

源代码包中的代理程序工程文件位于~\ net-snmp-5.2.1.2\win32\snmpd下，因为它可移植到多个平台，主程序代码中有许多平台开关，和各种选项，非常庞大。为了简化开发和调试，我们使用了一个帮助文档中介绍的简化的代理端程序框架。这个框架非常精简，在性能上可能比原版差很多，但用来测试和开发mib库就比较高效了。程序的运行机制：程序启动，载入初始化mib模块，然后进入一个等待呼叫的无限循环，代码片段如下：

```
… …
init_MyMib();
… …
    while(keep_running)
    {
  /* if you use select(), see snmp_select_info() in snmp_api(3) */
  /*    --- OR ---  */
  agent_check_and_process(1); /* 0 == don't block */
}
```

我们所要做的就是实现init_MyMib()函数，即自定义的mib模块。关于无限循环中的阻塞与否，根据mib模块而定。如果你的应用程序需要以非阻塞方式处理SNMP数据流，就使用一步接口（例如GUI、线程、forking等）。否则，只需要使用同步接口就可以了。在阻塞模式下，程序会占用大量cpu资源。具体实现，参见附录5.1。下面从mib库开始。

3.3 自定义MIB

这里我们定义了一个私人节点的MIB，位于节点.iso.org.dod.internet.private.enterprises.foxmail下，数字Oid为：.1.3.6.1.4.1.310。树型结构如下：

```
    foxmail                    .1.3.6.1.4.1.310
|----SecondCounter             .1.3.6.1.4.1.310.1
```

```
|----WeekTime                    .1.3.6.1.4.1.310.2
+----ExampleTable                .1.3.6.1.4.1.310.3
     +----ExampleEntry           .1.3.6.1.4.1.310.3.1
          |----UserIndex         .1.3.6.1.4.1.310.3.1.1
          |----UserStatus        .1.3.6.1.4.1.310.3.1.2
          |----CheckTime         .1.3.6.1.4.1.310.3.1.3
          |----MonSet            .1.3.6.1.4.1.310.3.1.4
```

在这个MIB中，定义了两个简单变量SecondCounter:整型，WeekTime:时间类型；还定义了一个表对象ExampleTable,其中UserIndex为索引：整型，UserStatus:字符串，CheckTime:时间类型，以上三个都是只读，MonSet为读写变量：整型。参见附录5.4

3.4 自定义MIB——简单变量的实现

把自定义的MIB命名为MyMib.txt,参照上面章节，放到net-snmp二进制安装路径~\usr\share\snmp\mibs下。配置snmp.conf文件，加入新的库MyMib.txt，语法为MIBS=+MyMib，这样才能让系统搜索到。然后在CMD提示符下输入：mib2c SecondCounter选择生成net-snmp版的代码。就会生成SecondCounter.c和SecondCounter.h文件，同样可以产生WeekTime.c和WeekTime.h文件。

由模版生成的文件，不论是简单变量还是表对象，其整体结构都是固定模式。在模版头文件中对节点进行宏定义，函数声明。在模版实现文件中，分为两大块：一是初始化函数，主要用来对变量注册；二是响应函数，用来响应管理端的查询命令，响应函数的返回值，就是我们要手工实现的。我们所要做的工作就是把数据以一种合理的方式导入到其中。

模版是针对单个变量来处理的：

1） 初始化：

```
init_SecondCounter(void)
{
   static oid SecondCounter_oid[] = { 1,3,6,1,4,1,310,1 };
DEBUGMSGTL(("SecondCounter", "Initializing\n"));
tsnmp_create_handler_registration("SecondCounter", handle_SecondCounter, SecondCounter_oid,
OID_LENGTH(SecondCounter_oid),HANDLER_CAN_RWRITE
                    ));
}
```

如果有多个变量就要注册多次。

2） 处理响应函数

```
handle_SecondCounter(netsnmp_mib_handler *handler,
                     netsnmp_handler_registration *reginfo,
```

```
                netsnmp_agent_request_info   *reqinfo,
                netsnmp_request_info         *requests)
```

其实现主要为：

```
 switch(reqinfo->mode) {
     case MODE_GET:
        snmp_set_var_typed_value(requests->requestvb, ASN_INTEGER,
pointer to the scalar's data */,
                                          /* XXX: the length of the data in bytes */);
```

如果有多个变量，就要简单重复switch语句，因为switch只提供了通信原语的选择，变量的选择依赖各自的处理句柄函数。

显然这两套代码如果合并起来就显得冗余了，因为很多地方可以共用一套代码。在这里参考源代码包~\net-snmp-5.2.1.2\agent\mibgroup\examples下的example.c和example.h文件合并我们的MIB。

主要实现方式是：

1）通过一种变量数组，把MIB中的变量列举出来，代码片断如下：

```
struct variable2 foxmail_variables[] =
{
    {FoxmailINT,ASN_INTEGER,RONLY,var_foxmail,1,{1}},
    {FoxmailTIMETICKS, ASN_TIMETICKS, RONLY, var_foxmail, 1, {2}}
  /*… … …加入其他的变量… … … */
};
```

结构体variable2的定义如下:

```
struct variable2 {
   u_char         magic;      /* passed to function as a hint */
   u_char         type;       /* type of variable */
   u_short        acl;        /* access control list for variable */
   FindVarMethod  *findVar;    /* function that finds variable */
   u_char         namelen;    /* length of name below */
   oid            name[2];    /* object identifier of variable */
};
```

通过上面代码，可以看出，数组对每个变量都一一说明了变量类型、节点位置、读写状态、实现函数等。

2） 然后再对变量数组进行注册初始化：

```
REGISTER_MIB("foxmail", foxmail_variables, variable2, foxmail_variables_oid);
```

3）从数组可知实现函数为var_foxmail();其原型为：

```
u_char * var_foxmail(struct variable *vp,
                     oid * name,
                     size_t * length,
                     int exact, size_t * var_len, WriteMethod ** write_method)
```

它通过一个switch语句来选择响应变量，代码片断如下：

```
    switch (vp->magic) {
  case FoxmailINT:     /*这里为数组中的索引值*/
     long_ret=XXX;    /*XXX为返回值，这里可以用指针来引入内部或外部数据*/
/* *write_method=write_foxmailINT  如果是可以set的变量，需调用写函数*/
     return (u_char *) & long_ret;
  case FoxmailTIMETICKS:
  … … …
        }
```

总之，这种方法比较紧凑有效，完整代码参见附录5.2

3.5 自定义MIB——表对象的实现

为了更加有效的描述对象，引入表来把一系列相关的信息进行综合。因此表就由具有一个或多个变量的列组成。从实现方式和数据结构来看，表是一个结构体变量数组；从系统管理来看，就是数据库中的表单。从管理数据库的需求出发，就要求实现表单的插入、删除、恢复等基本功能，更高级的功能还应实现多个表的依赖关系、触发机制等。在底层实现上这些都要通过操纵结构体数组来完成。

Mib2c提供了表的生成模版。其目的之一是可以降低对SNMP知识的要求，二是分离请求中对数据的定位和响应中对数据的处理，使得实现上集中在对数据的处理上。因此它产生的模版主要由三个部分组成：一是数据结构，对请求数据的构造；二是数据检索，对请求数据的定位；三是操纵数据，对请求数据进行GET或SET。

不同的需求有不同的实现方式，mib2c根据表中导入数据的来源提供了两大类模版。

其分类按照运行mib2c时的交互信息：一类为数据相对代理处于外部，适合于监视和操纵外部数据的MIB，如从操作系统或其他系统的接口提取信息；并且表中的行的创建销毁，通常都独立于SNMP代理。第二类是数据为代理所有，代理对数据的操纵直接反映到数据源。

本文档只针对第一种类型实现表，且称之为外部方式。在mib2c进入外部方式，仍有三个模版，分别为：mib2c.iterate.conf，mib2c.iterate_access.conf，mib2c.mfd.conf；前两种方式基本一样，使用了helpers中一种叫iterator的API，特点是可以处理无序数据，但是效率较低，适合于小型数据存储，对内存、响应时间要求不是很严格的应用；后一种就很复杂了，它提供了更加细致的代码框架，通过多层交互，可以区分实时、半实时、永久型的数据，可以选择数据的存储方式，可以做列或表之间的依赖关系，数据结构的自动绑定或手工编辑，以及其他复杂的选项。我们只实现前两种较为简单的。

3.5.1 mib.iterator.conf模版的实现

首先，这个模版生成了Example.c和Example.h文件。在Example.c文件中包含下列函数和结构体，形参省略：

struct ExampleTable_entry；

void init_ExampleTable;

void initialize_table_ExampleTable;

Netsnmp_Node_Handler ExampleTable_handler;

Netsnmp_First_Data_Point ExampleTable_get_first_data_point;

Netsnmp_Next_Data_Point ExampleTable_get_next_data_point;

struct ExampleTable_entry *ExampleTable_createEntry；

void ExampleTable_removeEntry；

实现过程我们按数据结构，数据检索，数据操纵来说明函数的调用和处理过程。

1） 结构体：

```
struct ExampleTable_entry {
  /* Index values */                /*这个结构体把exampleTable中的*/
  long UserIndex;                  /*变量都列举，并指明了索引变量 */
                                /*并有一个生成链表的next指针 */
  /* Column values */
 /*  long UserIndex;*/              /*去掉重复              */
  char UserStatus;
  u_long CheckTime;
  long MonSet;
  long old_MonSet;
```

```
    /* Illustrate using a simple linked list */
  /*    int   valid;*/                    /*没有用到的变量          */
    struct ExampleTable_entry *next;
  };
```

通过这个结构体，我们把从外部读入的数据做成一个链表，并把链表头地址传递给

ExampleTable_get_first_data_point;

ExampleTable_get_next_data_point;

用这两个函数遍历链表。

2） 数据的检索：

数据的检索由ExampleTable_handler函数来完成，其原型为：

```
int  ExampleTable_handler(
      netsnmp_mib_handler              *handler,
      netsnmp_handler_registration     *reginfo,
      netsnmp_agent_request_info       *reqinfo,
        netsnmp_request_info           *requests)
```

检索过程通过一个循环包含的两个语句：

```
for (request=requests; request; request=request->next) {
     table_entry = (struct ExampleTable_entry *)          /*检索匹配表中的行*/
           netsnmp_extract_iterator_context(request);
          table_info = netsnmp_extract_table_info(request);       /*检索匹配表中的列*/
```

这样相当于通过行号和列号确定了数据。而上面的两个函数需要调用

ExampleTable_get_first_data_point;

ExampleTable_get_next_data_point;

来遍历链表。

3） 数据的操纵：

数据的操纵涉及到对请求命令的响应，其代码片断如下：

```
switch (reqinfo->mode) {
case MODE_GET:
        switch (table_info->colnum) {
            case COLUMN_USERINDEX:
                snmp_set_var_typed_value( request->requestvb, ASN_INTEGER,
                                    table_entry->UserIndex,
                                    sizeof(table_entry->UserIndex));
            … … …          /*对于get命令，通过snmp_set_var_typed_value*/
        }                  /*返回变量值                          */
case MODE_SET_RESERVE1:       /*对于set命令，就涉及这六个步骤，这里省略了 */
case MODE_SET_RESERVE2:       /*具体实现代码，我们所做的程序中，没有涉及表 */
case MODE_SET_FREE:          /*中行的创建和删除，模版说明中认为，set 一个 */
case MODE_SET_ACTION:         /*不存在的索引，就可以在表中为其创建一行    */
case MODE_SET_UNDO:
case MODE_SET_COMMIT:
}                          /*代码已合并到mib.iterator_access.conf模版中*/
```

关于表的set操作的流程图，如下图：

图2. SET的执行过程

3.5.2 mib.iterator_access.conf模版的实现

首先，这个模版生成了 ExampleTable.c

ExampleTable.h

ExampleTable_columns.h

ExampleTable_enums.h

ExampleTable_checkfns_local.h

ExampleTable_checkfns_local.c

ExampleTable_checkfns.h

ExampleTable_checkfns.c

ExampleTable_access.h

ExampleTable_access.c

这些文件实际上是对mib.iterator.conf模版功能的细化，提供更加完善方式的控制。只有部分需要修改，根据自定义的mib，只编辑了ExampleTable.c，ExampleTable_access.c等文件。

它提供了get_XXX（）和set_XXX（）函数来完成数据的获取和设置，需要手工实现。数据的组织不再自动生成，可以自己以其他方式实现，但仍要能关联到遍历函数上。ExampleTable_checkfns_local.c，ExampleTable_checkfns.c文件都是对set的检查操作。在主体结构和运行过程上仍和mib.iterator.conf模版一样。参见附录5.3。

3.6 代码的合并

在实现整个mib时，只需把简单变量的初始化函数和表实现的初始化函数合并。在这里简单变量的初始化函数为：void init_foxmail。把表的初始化函数void initialize_table_ExampleTable放到init_foxmail内部，再将init_foxmail作为主函数的初始化函数，修改相关头文件的引用，代码的简单合并就完成了。最后以主函数建立工程编译全部文件。

3.7 配置和运行

在主函数中，代理的初始化由三个语句顺序完成：

```
init_agent("example-demon");  /*运行代理名*/

init_foxmail();

init_snmp("example-demon");  /*读入的配置文件名*/
```

这里需要写一个名为example-demon.conf的配置文件，配置方式同介绍的snmpd.conf配置一样。然后放在安装目录~\usr\etc\snmp下。关闭防火墙，在命令提示符下运行该程序，然后用snmpget,snmpgetnext,snmptable,snmpset测试程序。

4 开发中的问题与解决

在开始拿到这个软件包时，并不知道从哪里下手，以前没接触过这么大的代码包，只是按照帮助中关于win32平台下的说明往下做，但在编译源代码包时，编译到代理程序时就进行不下去了，出现了很多全局变量的依赖错误，当时的判断是编译环境的设置问题。于是转到CygWin下，利用自带的makefile来编译整个代码，非常成功的编译并安装到CygWin中。我们试着在其中编译了几个例子以及自己写的几个程序，也成功的通过了。在例子中，提供了一个makefile文件，于是就观察其环境设置。在其gcc的编译中，用到了

```
CFLAGS=-I. `net-snmp-config --cflags`

BUILDLIBS=`net-snmp-config --libs`

BUILDAGENTLIBS=`net-snmp-config --agent-libs`
```

查找配置文件，发现这里面涉及到四个关键库（见前面环境设置章节），在VC中我都没有加上去。在VC中添加上去后，变量依赖问题大大减少，发现遗留的错误是由socket引起的。加上wsock32.lib后，在VC下就可以通过先前的程序了。

但程序运行时有碰到了问题，查不到结果，或者程序崩溃。最后我们通过检查环境设置，对比文件，发现问题出在netsnmpagent.lib这个库上。在原有的基础上重编译这个库，新编译的库比原来大了几倍，也就是说库的编译存在一个先后顺序关系，可能这个库要引用到

其他库，可能VC没那么智能，需要手工设置。解决这个问题后，后面碰到的基本上是编写程序本身的错误了。

5 总结

net-snmp是一个功能很强的软件，这个开源项目仍在不断前进中。我们所做的只使用了其中的一小部分功能，主要是学习了它的一个开发流程、简单的结构框架，以及工具的使用；另外也学习了snmp协议，了解了开发包应为用户隔离底层细节，提供应用接口给用户。Net-snmp的帮助文档并不适合新手入门，它的帮助都是针对功能写的，一般通过它的开发网站中的tutorials来编译几个例子逐渐了解其功能，再回头查看帮助说明。Net-snmp项目目前放在www.sourceforge.net，可以加入net-snmp的邮件列表寻求帮助。

6 附录

6.1 主函数foxmail_new.c

```
/*主函数：foxmail_new.c */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include <signal.h>

#include "Display_time.h"
#include "ExampleTable.h"
static int keep_running;

RETSIGTYPE
stop_server(int a) {
    keep_running = 0;
}

int
main (int argc, char **argv) {
  int agentx_subagent=0; /* change this if you want to be a SNMP master agent */
  int background = 0; /* change this if you want to run in the background */
  int syslog = 0; /* change this if you want to use syslog */

  /* print log errors to syslog or stderr */
  if (syslog)
    snmp_enable_calllog();
  else
```

```
    snmp_enable_stderrlog();

  /* we're an agentx subagent? */
  if (agentx_subagent) {
    /* make us a agentx client. */
    netsnmp_ds_set_boolean(NETSNMP_DS_APPLICATION_ID,
NETSNMP_DS_AGENT_ROLE, 1);
  }

  /* run in background, if requested */
  if (background && netsnmp_daemonize(1, !syslog))
      exit(1);

  /* Initialize tcpip, if necessary */
  SOCK_STARTUP;

  /* Initialize the agent library */
  init_agent("example-demon");

  /* Initialize our mib code here */
       init_foxmail();

  /* initialize vacm/usm access control  */
  if (!agentx_subagent) {
    void  init_vacm_vars();
    void  init_usmUser();
  }

  /* Example-demon will be used to read example-demon.conf files. */
```

```
  init_snmp("example-demon");

  /* If we're going to be a snmp master agent, initial the ports */
  if (!agentx_subagent)
    init_master_agent();  /* open the port to listen on (defaults to udp:161) */

  /* In case we recevie a request to stop (kill -TERM or kill -INT) */
  keep_running = 1;
  signal(SIGTERM, stop_server);
  signal(SIGINT, stop_server);
  snmp_log(LOG_INFO,"example-demon is up and running.\n");

  /* your main loop here... */
  while(keep_running) {
    /* if you use select(), see snmp_select_info() in snmp_api(3) */
    /*     --- OR ---  */
    agent_check_and_process(1); /* 0 == don't block */
  }

  /* at shutdown time */
  snmp_shutdown("example-demon");
  SOCK_CLEANUP;

  return 0;
}
```

6.2 简单变量实现代码

6.2.1 display_time.c

```
    /*display_time.c*/
```

```
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>

#if HAVE_STDLIB_H
#include <stdlib.h>
#endif

#if TIME_WITH_SYS_TIME
# ifdef WIN32
#  include <sys/timeb.h>
# else
#  include <sys/time.h>
# endif
# include <time.h>
#else
# if HAVE_SYS_TIME_H
#  include <sys/time.h>
# else
#  include <time.h>
# endif
#endif

#include "util_funcs.h"

#include "Display_time.h"
#include "ExampleTable.h"

struct variable2 foxmail_variables[] =
```

```
{
    {FoxmailINT,ASN_INTEGER,RONLY,var_foxmail,1,{1}},
    {FoxmailTIMETICKS, ASN_TIMETICKS, RONLY, var_foxmail, 1, {2}}
};

oid foxmail_variables_oid[] = { 1,3,6,1,4,1,310 };

void
init_foxmail(void)
{
   REGISTER_MIB("foxmail", foxmail_variables, variable2,
            foxmail_variables_oid);
   initialize_table_ExampleTable();
}

u_char        *
var_foxmail(struct variable *vp,
          oid * name,
          size_t * length,
          int exact, size_t * var_len, WriteMethod ** write_method)
{
   static long     long_ret;   /* for everything else */
   static time_t timep;
   struct tm *p;
   time(&timep);
  p=gmtime(&timep);
   DEBUGMSGTL(("foxmail", "var_foxmail entered\n"));
   if (header_generic(vp, name, length, exact, var_len, write_method) ==
       MATCH_FAILED)
```

```
        return NULL;

    switch (vp->magic) {



    case FoxmailINT:
        long_ret=timep;
        return (u_char *) & long_ret;

    case FoxmailTIMETICKS:
            time(&timep);
            long_ret=(p->tm_wday*24*3600+(p->tm_hour+8)*3600+p->tm_min*60+p->tm_sec)
*100;
            return (u_char*) & long_ret;
    default:

        DEBUGMSGTL(("snmpd", "unknown sub-id %d in examples/var_example\n",
                vp->magic));
    }

    return NULL;
}
```

6.2.2 display_time.h

```
/*display_time.h*/
#ifndef DISPLAY_TIME_H
#define DISPLAY_TIME_H

config_require(util_funcs)
extern void init_foxmail(void);
```

```
extern FindVarMethod var_foxmail;

#define FoxmailINT 1
#define FoxmailTIMETICKS 2

#endif
```

6.3 表的实现

6.3.1 ExampleTable.c

```
/*ExampleTable.c*/
/*
 * Note: this file originally auto-generated by mib2c using
 *       : mib2c.iterate_access.conf,v 1.11 2004/08/31 10:23:16 dts12 Exp $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
#include "ExampleTable.h"
#include "ExampleTable_checkfns.h"
#include "ExampleTable_access.h"

static netsnmp_oid_stash_node *undoStorage = NULL;
static netsnmp_oid_stash_node *commitStorage = NULL;

struct undoInfo {
  void *ptr;
  size_t len;
};
```

```
struct commitInfo {
  void *data_context;
  int have_committed;
  int new_row;
};

void
ExampleTable_free_undoInfo(void *vptr) {
   struct undoInfo *ui = vptr;
   if (!ui)
      return;
   SNMP_FREE(ui->ptr);
   SNMP_FREE(ui);
}

/** Initialize the ExampleTable table by defining its contents and how it's structured */
void
initialize_table_ExampleTable(void)
{
   static oid ExampleTable_oid[] = {1,3,6,1,4,1,310,3};
   netsnmp_table_registration_info *table_info;
   netsnmp_handler_registration *my_handler;
   netsnmp_iterator_info *iinfo;

   /** create the table registration information structures */
   table_info = SNMP_MALLOC_TYPEDEF(netsnmp_table_registration_info);
   iinfo = SNMP_MALLOC_TYPEDEF(netsnmp_iterator_info);
```

```
my_handler = netsnmp_create_handler_registration("ExampleTable",
                                ExampleTable_handler,
                                ExampleTable_oid,
                                OID_LENGTH(ExampleTable_oid),
                                HANDLER_CAN_RWRITE
                                );

if (!my_handler || !table_info || !iinfo) {
   snmp_log(LOG_ERR, "malloc failed in initialize_table_ExampleTable");
   return; /** Serious error. */
  }

   /**************************************************
 * Setting up the table's definition
 */
netsnmp_table_helper_add_indexes(table_info,
                        ASN_INTEGER, /** index: MachineNumber */
                    0);

/** Define the minimum and maximum accessible columns.  This
   optimizes retrival. */
table_info->min_column = 1;
table_info->max_column = 4;

/** iterator access routines */
iinfo->get_first_data_point = ExampleTable_get_first_data_point;
iinfo->get_next_data_point = ExampleTable_get_next_data_point;

/** you may wish to set these as well */
```

```
#ifdef MAYBE_USE_THESE
    iinfo->make_data_context = ExampleTable_context_convert_function;
    iinfo->free_data_context = ExampleTable_data_free;

    /** pick *only* one of these if you use them */
    iinfo->free_loop_context = ExampleTable_loop_free;
    iinfo->free_loop_context_at_end = ExampleTable_loop_free;
#endif

    /** tie the two structures together */
    iinfo->table_reginfo = table_info;

    /**************************************************
     * registering the table with the master agent
     */
    DEBUGMSGTL(("initialize_table_ExampleTable",
                "Registering table ExampleTable as a table iterator\n"));
    netsnmp_register_table_iterator(my_handler, iinfo);
}

/** Initializes the ExampleTable module */
/*void
init_ExampleTable(void)
{
*/
  /** here we initialize all the tables we're planning on supporting */
/*    initialize_table_ExampleTable();
}*/
```

```
/** handles requests for the ExampleTable table, if anything else needs to be done */
int
ExampleTable_handler(
    netsnmp_mib_handler               *handler,
    netsnmp_handler_registration      *reginfo,
    netsnmp_agent_request_info        *reqinfo,
    netsnmp_request_info              *requests) {

    netsnmp_request_info *request;
    netsnmp_table_request_info *table_info;
    netsnmp_variable_list *var;
    struct commitInfo *ci = NULL;

    void *data_context = NULL;

    oid *suffix;
    size_t suffix_len;

    /** column and row index encoded portion */
    suffix = requests->requestvb->name + reginfo->rootoid_len + 1;
    suffix_len = requests->requestvb->name_length -
       (reginfo->rootoid_len + 1);

    for(request = requests; request; request = request->next) {
       var = request->requestvb;
       if (request->processed != 0)
          continue;

       switch (reqinfo->mode) {
```

```
    case MODE_GET:
        data_context =  netsnmp_extract_iterator_context(request);
        if (data_context == NULL) {
            netsnmp_set_request_error(reqinfo, request,
                                SNMP_NOSUCHINSTANCE);
            continue;
        }
        break;

    case MODE_SET_RESERVE1:
        data_context =  netsnmp_extract_iterator_context(request);
        if (data_context == NULL) {
            netsnmp_set_request_error(reqinfo, request,
                                SNMP_ERR_NOCREATION);
            continue;
        }
        break;

    default: /* == the other SET modes */
        ci = netsnmp_oid_stash_get_data(commitStorage,
                                 suffix+1, suffix_len-1);
        break;
    }

    /** extracts the information about the table from the request */
    table_info = netsnmp_extract_table_info(request);
    /** table_info->colnum contains the column number requested */
    /** table_info->indexes contains a linked list of snmp variable
      bindings for the indexes of the table.  Values in the list
```

```
    have been set corresponding to the indexes of the
    request */
  if (table_info == NULL) {
     continue;
  }

  switch(reqinfo->mode) {
     case MODE_GET:
        switch(table_info->colnum) {
           case COLUMN_MACHINENUMBER:
                 {
                    long *retval;
                    size_t retval_len = 0;
                    retval = get_MachineNumber(data_context, &retval_len);
                    if (retval)
                       snmp_set_var_typed_value(var, ASN_INTEGER,
                                 (const u_char *) retval,
                                 retval_len);
                 }
              break;

           case COLUMN_MACHINESTATUS:
                 {
                    char *retval;
                    size_t retval_len = 0;
                    retval = get_MachineStatus(data_context, &retval_len);
                    if (retval)
                       snmp_set_var_typed_value(var, ASN_OCTET_STR,
                                 (const u_char *) retval,
```

```
                        retval_len);
        }
    break;

case COLUMN_CHECKTIME:
        {
            u_long *retval;
            size_t retval_len = 0;
            retval = get_CheckTime(data_context, &retval_len);
            if (retval)
                snmp_set_var_typed_value(var, ASN_TIMETICKS,
                        (const u_char *) retval,
                        retval_len);
        }
    break;

case COLUMN_MONSET:
        {
            long *retval;
            size_t retval_len = 0;
            retval = get_MonSet(data_context, &retval_len);
            if (retval)
                snmp_set_var_typed_value(var, ASN_INTEGER,
                        (const u_char *) retval,
                        retval_len);
        }
    break;

default:
```

```
        /** We shouldn't get here */
            snmp_log(LOG_ERR, "problem encountered in ExampleTable_handler: unknown
column\n");
        }
        break;

    case MODE_SET_RESERVE1:
        ci = netsnmp_oid_stash_get_data(commitStorage,
                                suffix+1, suffix_len-1);

        if (!ci) {
            /** create the commit storage info */
            ci = SNMP_MALLOC_STRUCT(commitInfo);
            if (!data_context) {
              // ci->data_context = ExampleTable_create_data_context(table_info->indexes);
               ci->new_row = 1;
            } else {
               ci->data_context = data_context;
            }
            netsnmp_oid_stash_add_data(&commitStorage,
                                suffix+1, suffix_len-1, ci);
        }
    break;

    case MODE_SET_RESERVE2:
        switch(table_info->colnum) {
              case COLUMN_MONSET:
                {
                   long *retval;
```

```
        size_t retval_len = 0;
        struct undoInfo *ui = NULL;
        int ret;

    /** first, get the old value */
        retval = get_MonSet(ci->data_context, &retval_len);
        if (retval) {
            ui = SNMP_MALLOC_STRUCT(undoInfo);
            ui->len = retval_len;
            memdup((u_char **) &ui->ptr,
                   (u_char *) retval,
                   ui->len);
        }

    /** check the new value, possibly against the
       older value for a valid state transition */
        ret = check_MonSet(request->requestvb->type,
                           (long *) request->requestvb->val.integer,
                           request->requestvb->val_len,
                           retval, retval_len);
        if (ret != 0) {
            netsnmp_set_request_error(reqinfo, request,
                                      ret);
            ExampleTable_free_undoInfo(ui);
        } else if (ui) {
       /** remember information for undo purposes later */
            netsnmp_oid_stash_add_data(&undoStorage,
                                       suffix,
                                       suffix_len,
```

```
                                        ui);
                }


            }
            break;
        default:
            netsnmp_set_request_error(reqinfo, request,
                                SNMP_ERR_NOTWRITABLE);
            break;
    }
    break;


case MODE_SET_ACTION:
/** save a variable copy */
    switch(table_info->colnum) {
            case COLUMN_MONSET:
            {
                int ret;
                ret = set_MonSet(ci->data_context,
                        (long *) request->requestvb->val.integer,
                        request->requestvb->val_len);
                if (ret) {
                    netsnmp_set_request_error(reqinfo, request,
                                        ret);
                }
            }
            break;
    }
    break;
```

```
case MODE_SET_COMMIT:
   if (!ci->have_committed) {
      /** do this once per row only */
      ExampleTable_commit_row(&ci->data_context, ci->new_row);
      ci->have_committed = 1;
   }
   break;

case MODE_SET_UNDO:
 /** save a variable copy */
   switch(table_info->colnum) {
          case COLUMN_MONSET:
           {
              int retval;
              struct undoInfo *ui;
              ui = netsnmp_oid_stash_get_data(undoStorage,
                                    suffix,
                                    suffix_len);
              retval = set_MonSet(ci->data_context, ui->ptr,
                        ui->len);
              if (retval) {
                 netsnmp_set_request_error(reqinfo, request,
                                SNMP_ERR_UNDOFAILED);
              }
           }
           break;
   }
   break;
```

```
        case MODE_SET_FREE:
            break;

        default:
            snmp_log(LOG_ERR, "problem encountered in ExampleTable_handler: unsupported
mode\n");
        }
    }

    /** clean up after all requset processing has ended */
    switch(reqinfo->mode) {
    case MODE_SET_UNDO:
    case MODE_SET_FREE:
    case MODE_SET_COMMIT:
        /** clear out the undo cache */
        netsnmp_oid_stash_free(&undoStorage, ExampleTable_free_undoInfo);
        netsnmp_oid_stash_free(&commitStorage, netsnmp_oid_stash_no_free);
    }

    return SNMP_ERR_NOERROR;
}
```

6.3.2 ExampleTable.h

```
/*ExampleTable.h*/
/*
 * Note: this file originally auto-generated by mib2c using
 *       : mib2c.iterate_access.conf,v 1.11 2004/08/31 10:23:16 dts12 Exp $
 */
#ifndef EXAMPLETABLE_H
```

```
#define EXAMPLETABLE_H

/** other required module components */
config_require(ExampleTable_access)
config_require(ExampleTable_checkfns)

/* function declarations */
void init_ExampleTable(void);
void initialize_table_ExampleTable(void);
Netsnmp_Node_Handler ExampleTable_handler;



/* column number definitions for table ExampleTable */
#include "ExampleTable_columns.h"

/* enum definions */
#include "ExampleTable_enums.h"

#endif /** EXAMPLETABLE_H */
```

6.3.3 ExampleTable_access.c

```
/*ExampleTable_access.c*/
/*
 * Note: this file originally auto-generated by mib2c using
 *       : mib2c.access_functions.conf,v 1.9 2004/10/14 12:57:33 dts12 Exp $
 */

#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include <net-snmp/agent/net-snmp-agent-includes.h>
```

```
#include "ExampleTable_access.h"
#include "ExampleTable_enums.h"
#include <stdio.h>

struct ExampleTable_entry {
    long MachineNumber;
    char MachineStatus[10];
    u_long CheckTime;
    long   MonSet;
    struct ExampleTable_entry *next;
};

struct ExampleTable_entry  *ExampleTable_head=NULL;

/* create a new row in the (unsorted) table */
struct ExampleTable_entry *
ExampleTable_createEntry(long  MachineNumber,
                         char *MachineStatus,
                         u_long CheckTime,
                         long MonSet
                                 )
 {
    struct ExampleTable_entry *entry;

    entry = SNMP_MALLOC_TYPEDEF(struct ExampleTable_entry);
    if (!entry)
        return NULL;

    entry->MachineNumber = MachineNumber;
```

```
    strcpy(entry->MachineStatus ,MachineStatus);
    entry->CheckTime     = CheckTime;
      entry->MonSet       =MonSet;
    entry->next          = ExampleTable_head;
    ExampleTable_head    = entry;

    return entry;
}

/* remove a row from the table */
void
ExampleTable_removeEntry( struct ExampleTable_entry *entry )
{
    struct ExampleTable_entry *ptr, *prev;

    if (!entry)
        return;    /* Nothing to remove */

    for ( ptr  = ExampleTable_head, prev = NULL;
          ptr != NULL;
          prev = ptr, ptr = ptr->next ) {
        if ( ptr == entry )
            break;
    }
    if ( !ptr )
        return;    /* Can't find it */

    if ( prev == NULL )
        ExampleTable_head = ptr->next;
```

```
    else
        prev->next = ptr->next;

    SNMP_FREE( entry );   /* XXX - release any other internal resources */
}

void data_read(void)
{
     FILE *fp;
  int i;
  struct  ExampleTable_entry *temp;
     fp=fopen("data.txt","r");
     while (ExampleTable_head)/*clean link list begin*/
           {
                temp=ExampleTable_head->next;
                ExampleTable_removeEntry(ExampleTable_head);
                ExampleTable_head=temp;
           }/*clean link list end*/

      temp=SNMP_MALLOC_TYPEDEF(struct ExampleTable_entry);
      temp->next=NULL;

    /*set up a link list begin*/
      if(fp)
      {
  i=fscanf(fp,"%d %s %d %d",&temp->MachineNumber, temp->MachineStatus, &temp-
>CheckTime,&temp->MonSet);
  /*fscanf return reading var numbers .if EOF return -1.feof() dosen't work well*/
      while(i>0)
```

```
  {
        ExampleTable_createEntry(temp->MachineNumber,temp->MachineStatus,temp->CheckTime,temp->MonSet);
        i=fscanf(fp,"%d %s %d %d",&temp->MachineNumber, temp->MachineStatus, &temp->CheckTime,&temp->MonSet);
    }
SNMP_FREE(temp);
fclose(fp);/*set up a link list end*/
    }
    else
    {
        printf("Error:Can't open data.txt!\n");
        /*exit(1);*/
    }
}
```

```
static u_long long_ret;
/** returns the first data point within the ExampleTable table data.

   Set the my_loop_context variable to the first data point structure
   of your choice (from which you can find the next one).  This could
   be anything from the first node in a linked list, to an integer
   pointer containing the beginning of an array variable.

   Set the my_data_context variable to something to be returned to
   you later that will provide you with the data to return in a given
   row.  This could be the same pointer as what my_loop_context is
   set to, or something different.
```

```
    The put_index_data variable contains a list of snmp variable
    bindings, one for each index in your table.  Set the values of
    each appropriately according to the data matching the first row
    and return the put_index_data variable at the end of the function.
*/
netsnmp_variable_list *
ExampleTable_get_first_data_point(void **my_loop_context, void **my_data_context,
                    netsnmp_variable_list *put_index_data,
                    netsnmp_iterator_info *mydata)
{
    data_read();
    *my_loop_context = ExampleTable_head;
    *my_data_context = ExampleTable_head;

return ExampleTable_get_next_data_point(my_loop_context, my_data_context,
                            put_index_data,  mydata );
}

/** functionally the same as ExampleTable_get_first_data_point, but
   my_loop_context has already been set to a previous value and should
   be updated to the next in the list.  For example, if it was a
   linked list, you might want to cast it to your local data type and
   then return my_loop_context->next.  The my_data_context pointer
   should be set to something you need later and the indexes in
   put_index_data updated again. */
netsnmp_variable_list *
ExampleTable_get_next_data_point(void **my_loop_context, void **my_data_context,
                    netsnmp_variable_list *put_index_data,
                    netsnmp_iterator_info *mydata)
```

```
{
      struct ExampleTable_entry *entry = (struct ExampleTable_entry *)*my_loop_context;
    netsnmp_variable_list *idx = put_index_data;


    if ( entry )
          {
       snmp_set_var_value( idx, (u_char *)&entry->MachineNumber, sizeof(entry-
>MachineNumber) );
       idx = idx->next_variable;
       *my_data_context = (void *)entry;
       *my_loop_context = (struct ExampleTable_entry *)entry->next;
         }
         else
               {
       return NULL;
               }
    return put_index_data;


}


/** Create a data_context for non-existent rows that SETs are performed on.
 *  return a void * pointer which will be passed to subsequent get_XXX
 *  and set_XXX functions for data retrival and modification during
 *  this SET request.
 *
 *  The indexes are encoded (in order) into the index_data pointer,
 *  and the column object which triggered the row creation is available
 *  via the column parameter, if it would be helpful to use that information.
 */
```

```
/*
void *
ExampleTable_create_data_context(netsnmp_variable_list *index_data) {

    return entry; /* XXX: you likely want to return a real pointer */
/*
}
*/
/** If the implemented set_* functions don't operate directly on the
   real-live data (which is actually recommended), then this function
   can be used to take a given my_data_context pointer and "commit" it
   to whereever the modified data needs to be put back to.  For
   example, if this was a routing table you could publish the modified
   routes back into the kernel at this point.

   new_or_del will be set to 1 if new, or -1 if it should be deleted
   or 0 if it is just a modification of an existing row.

   If you free the data yourself, make sure to *my_data_context = NULL */
int
ExampleTable_commit_row(void **my_data_context, int new_or_del)
{
    /** Add any necessary commit code here */
    /*  */

    /* return no errors.  And there shouldn't be any!!!  Ever!!!  You
    should have checked the values long before this. */
    return SNMP_ERR_NOERROR;
}
```

```
/* User-defined data access functions (per column) for data in table ExampleTable */
/*
 * NOTE:
 * - these get_ routines MUST return data that will not be freed (ie,
 *   use static variables or persistent data).  It will be copied, if
 *   needed, immediately after the get_ routine has been called.
 * - these SET routines must copy the incoming data and can not take
 *   ownership of the memory passed in by the val pointer.
 */
/** XXX: return a data pointer to the data for the MachineNumber column and set
        ret_len to its proper size in bytes. */
      long *get_MachineNumber(void *data_context, size_t *ret_len) {
         struct ExampleTable_entry *entry=(struct ExampleTable_entry *)data_context;
         long_ret=entry->MachineNumber;
         *ret_len=sizeof(long_ret);
       return &long_ret; /** XXX: replace this with a pointer to a real value */
      }
/** XXX: return a data pointer to the data for the MachineStatus column and set
        ret_len to its proper size in bytes. */
      char *get_MachineStatus(void *data_context, size_t *ret_len) {
       struct ExampleTable_entry *entry=(struct ExampleTable_entry *)data_context;

         *ret_len=strlen(entry->MachineStatus);
       return entry->MachineStatus; /** XXX: replace this with a pointer to a real value */
      }
/** XXX: return a data pointer to the data for the CheckTime column and set
        ret_len to its proper size in bytes. */
```

```
u_long *get_CheckTime(void *data_context, size_t *ret_len) {
  struct ExampleTable_entry *entry=(struct ExampleTable_entry *)data_context;
    long_ret=entry->CheckTime;
    *ret_len=sizeof(long_ret);
  return &long_ret; /** XXX: replace this with a pointer to a real value */
}
/** XXX: return a data pointer to the data for the MonSet column and set
     ret_len to its proper size in bytes. */
long *get_MonSet(void *data_context, size_t *ret_len) {
  struct ExampleTable_entry *entry=(struct ExampleTable_entry *)data_context;
    long_ret=entry->MonSet;
    *ret_len=sizeof(long_ret);
  return &long_ret; /** XXX: replace this with a pointer to a real value */
}
/** XXX: Set the value of the MonSet column and return
     SNMP_ERR_NOERROR on success
     SNMP_ERR_XXX     for SNMP deterministic error codes
     SNMP_ERR_GENERR  on generic failures (a last result response). */
int set_MonSet(void *data_context, long *val, size_t val_len) {
        FILE *fp;
        struct ExampleTable_entry *temp=ExampleTable_head,*entry=data_context;
         memcpy(&entry->MonSet, val, val_len);
         fp=fopen("data.txt","w+");
          if(!fp)
            {DEBUGMSGTL(("set_MonSet","Open file failure\n"));
            return SNMP_ERR_NOACCESS;
            }
      else
          while(temp)
```

```
        {
            fprintf(fp,"\n%d %s %d %d",temp->MachineNumber,
                        temp->MachineStatus,
                                    temp->CheckTime,
                                    temp->MonSet);
            temp=temp->next;
        }
      fclose(fp);
    return SNMP_ERR_NOERROR;  /** XXX: change if an error occurs */
  }
```

6.3.4 ExampleTable_access.h

```
/*ExampleTable_access.h*/
/*
 * Note: this file originally auto-generated by mib2c using
 *       : mib2c.access_functions.conf,v 1.9 2004/10/14 12:57:33 dts12 Exp $
 */
#ifndef EXAMPLETABLE_ACCESS_H
#define EXAMPLETABLE_ACCESS_H

/** User-defined data access functions for data in table ExampleTable */
/** row level accessors */
Netsnmp_First_Data_Point  ExampleTable_get_first_data_point;
Netsnmp_Next_Data_Point   ExampleTable_get_next_data_point;
int ExampleTable_commit_row(void **my_data_context, int new_or_del);
void * ExampleTable_create_data_context(netsnmp_variable_list *index_data, int column);

/** column accessors */
    long *get_MachineNumber(void *data_context, size_t *ret_len);
```

```
    char *get_MachineStatus(void *data_context, size_t *ret_len);
    u_long *get_CheckTime(void *data_context, size_t *ret_len);
    long *get_MonSet(void *data_context, size_t *ret_len);
    int set_MonSet(void *data_context, long *val, size_t val_len);

#endif /* EXAMPLETABLE_ACCESS_H */
```

6.3.5 ExampleTable_checkfns.c

```
/*ExampleTable_checkfns.c*/
/*
 * Note: this file originally auto-generated by mib2c using
 *       : mib2c.check_values.conf,v 1.8 2004/01/12 00:43:45 rstory Exp $
 */

/**********************************************************************
 *                  NOTE   NOTE   NOTE
 *   This file is auto-generated and SHOULD NOT BE EDITED by hand.
 *   Modify the ExampleTable_checkfns_local.[ch] files insead so that you
 *   can regenerate this one as mib2c improvements are made.
 **********************************************************************/

/* standard headers */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include "ExampleTable_checkfns.h"
#include "ExampleTable_checkfns_local.h"
#include "ExampleTable_enums.h"

/** Decides if an incoming value for the MonSet mib node is legal.
```

```
 *  @param type    The incoming data type.
 *  @param val     The value to be checked.
 *  @param val_len The length of data stored in val (in bytes).
 *  @return 0 if the incoming value is legal, an SNMP error code otherwise.
 */
   int
   check_MonSet(int type, long *val, size_t val_len,
           long *old_val, size_t old_val_len) {

   int ret;

   /** Check to see that we were called legally */
     if (!val)
       return SNMP_ERR_GENERR;

   /** Check the incoming type for correctness */
     if (type != ASN_INTEGER)
       return SNMP_ERR_WRONGTYPE;

      ret = SNMP_ERR_NOERROR;


   /** looks ok, call the local version of the same function. */
     return check_MonSet_local(type, val, val_len, old_val, old_val_len);
   }
```

6.3.6 ExampleTable_checkfns.h

```
/*ExampleTable_checkfns.h*/
/*
 * Note: this file originally auto-generated by mib2c using
```

```
 *       : mib2c.iterate.conf,v 5.6 2003/02/20 00:52:07 hardaker Exp $
 */

/**********************************************************************
 *   This file is auto-generated and SHOULD NOT BE EDITED by hand.
 *   Modify the ExampleTable_checkfns_local.[ch] files insead.
 *   (so that you can regenerate this one as mib2c improvements are made)
 **********************************************************************/
#ifndef EXAMPLETABLE_CHECKFNS_H
#define EXAMPLETABLE_CHECKFNS_H

/** make sure we load the functions that you can modify */
config_require(ExampleTable_checkfns_local)

/* these functions are designed to check incoming values for
columns in the ExampleTable table for legality with respect to
datatype and value.
 */

    int check_MonSet(int type, long *val, size_t val_len, long *old_val, size_t old_val_len);

#endif /* EXAMPLETABLE_CHECKFNS_H */
```

6.3.7 ExampleTable_checkfns_local.c

```
/*ExampleTable_checkfns_local.c*/
/*
 * Note: this file originally auto-generated by mib2c using
 *       : mib2c.check_values_local.conf,v 5.2 2004/05/04 23:34:56 hardaker Exp $
 */
```

```
/* standard headers */
#include <net-snmp/net-snmp-config.h>
#include <net-snmp/net-snmp-includes.h>
#include "ExampleTable_checkfns.h"
#include "ExampleTable_enums.h"

/** Decides if an incoming value for the MonSet mib node is legal, from a local implementation specific viewpoint.
 *  @param type    The incoming data type.
 *  @param val     The value to be checked.
 *  @param val_len The length of data stored in val (in bytes).
 *  @return 0 if the incoming value is legal, an SNMP error code otherwise.
 */
   int
   check_MonSet_local(int type, long *val, size_t val_len, long *old_val, size_t old_val_len) {

   /** XXX: you may want to check aspects of the new value that
      were not covered by the automatic checks by the parent function. */

   /** XXX: you make want to check that the requested change from
       the old value to the new value is legal (ie, the transistion
       from one value to another is legal */

   /** if everything looks ok, return SNMP_ERR_NOERROR */
     return SNMP_ERR_NOERROR;
   }
```

6.3.8 ExampleTable_checkfns_local.h

```
/*ExampleTable_checkfns_local.h*/
```

```
/*
 * Note: this file originally auto-generated by mib2c using
 *        : : mib2c.check_values_local.conf,v 5.2 2004/05/04 23:34:56 hardaker Exp $
 *
 */
#ifndef EXAMPLETABLE_CHECKFNS_H
#define EXAMPLETABLE_CHECKFNS_H

/* these functions are designed to check incoming values for
columns in the ExampleTable table for legality with respect to
datatype and value according to local conventions.  You should modify
them as appropriate.  They will be called from parent check_value
functions that are auto-generated using mib2c and the parent functions
should NOT be modified.
 */

   int check_MonSet_local(int type, long *val, size_t val_len, long *old_val, size_t old_val_len);

#endif /* EXAMPLETABLE_CHECKFNS_H */
```

6.3.9 ExampleTable_columns.h

```
/* ExampleTable_columns.h*/
/*
 * Note: this file originally auto-generated by mib2c using
 *  : mib2c.column_defines.conf,v 5.1 2002/05/08 05:42:47 hardaker Exp $
 */
#ifndef EXAMPLETABLE_COLUMNS_H
#define EXAMPLETABLE_COLUMNS_H
```

```
/* column number definitions for table ExampleTable */
    #define COLUMN_MACHINENUMBER       1
    #define COLUMN_MACHINESTATUS       2
    #define COLUMN_CHECKTIME           3
    #define COLUMN_MONSET        4
#endif /* EXAMPLETABLE_COLUMNS_H */
```

6.3.10 ExampleTable_enums.h

```
/* ExampleTable_enums.h*/
/*
 * Note: this file originally auto-generated by mib2c using
 *  : mib2c.column_enums.conf,v 5.2 2003/02/22 04:09:25 hardaker Exp $
 */
#ifndef EXAMPLETABLE_ENUMS_H
#define EXAMPLETABLE_ENUMS_H

#endif /* EXAMPLETABLE_ENUMS_H */
```

6.4 自定义mib文件MyMib.txt

```
MyMIB DEFINITIONS::=BEGIN
    IMPORTS
        enterprises,OBJECT-TYPE,Integer32,TimeTicks
            FROM SNMPv2-SMI
    TEXTUAL-CONVENTION,  DisplayString FROM SNMPv2-TC;
    foxmail OBJECT IDENTIFIER::={enterprises 310}
    SecondCounter OBJECT-TYPE
        SYNTAX Integer32
        ACCESS read-write
        STATUS mandatory
         DESCRIPTION "This is a one minute counter from year 1970.1.1.0:0 to now"
```

```
        ::={foxmail 1}

    WeekTime OBJECT-TYPE
        SYNTAX TimeTicks
        ACCESS read-only
        STATUS mandatory
         DESCRIPTION "Recording taday's time and the day sorts in the week"
        ::={foxmail 2}

    ExampleTable       OBJECT-TYPE
        SYNTAX SEQUENCE OF ExampleEntry
        MAX-ACCESS  not-accessible
        STATUS current
        DESCRIPTION
       "A list of interface entries.  The number of entries is
       given by the value of ExampleNumber."
            ::= { foxmail 3 }

    ExampleEntry OBJECT-TYPE
            SYNTAX      ExampleEntry
            MAX-ACCESS  not-accessible
            STATUS      current
            DESCRIPTION
       "An entry containing management information applicable to a
       particular interface."
            INDEX   { UserIndex }
            ::= { ExampleTable 1 }

    ExampleEntry ::=
```

```
        SEQUENCE {
                UserIndex          InterfaceIndex,
                UserStatus         DisplayString,
                CheckTime              TimeTicks,
               MonSet               Integer32
               }

InterfaceIndex ::= TEXTUAL-CONVENTION
 DISPLAY-HINT "d"
 STATUS       current
 DESCRIPTION
      "A unique value, greater than zero, for each interface or
      interface sub-layer in the managed system.  It is
      recommended that values are assigned contiguously starting
      from 1.  The value for each interface sub-layer must remain
      constant at least from one re-initialization of the entity's
      network management system to the next re-initialization."
 SYNTAX       Integer32 (1..2147483647)


  UserIndex OBJECT-TYPE
          SYNTAX      InterfaceIndex
       MAX-ACCESS  read-only
       STATUS      current
          DESCRIPTION
      "A unique value, greater than zero, for each interface.  It
      is recommended that values are assigned contiguously
      starting from 1.  The value for each interface sub-layer
      must remain constant at least from one re-initialization of
```

```
        the entity's network management system to the next re-
        initialization."
            ::= { ExampleEntry 1 }

    UserStatus OBJECT-TYPE
            SYNTAX      DisplayString
            MAX-ACCESS  read-only
            STATUS      current
            DESCRIPTION
        "machine status ."
            ::= { ExampleEntry 2 }

    CheckTime OBJECT-TYPE
            SYNTAX      TimeTicks
            MAX-ACCESS  read-only
            STATUS      current
            DESCRIPTION
        "machine status checking time."
            ::= { ExampleEntry 3 }

    MonSet OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
       "An example to use set function."
        ::={ ExampleEntry 4}
END
```

发表于 @ 2010年01月11日 14:29:00 | 评论(loading...) | 举报| *收藏*

旧一篇:SNMP 对网络监控的作用 | 新一篇:用NET-SNMP软件包开发简单客户端代理

查看最新精华文章 请访问博客首页相关文章

◦ 发表评论

表 情:

评论内容:

用 户名:

发表评论

.