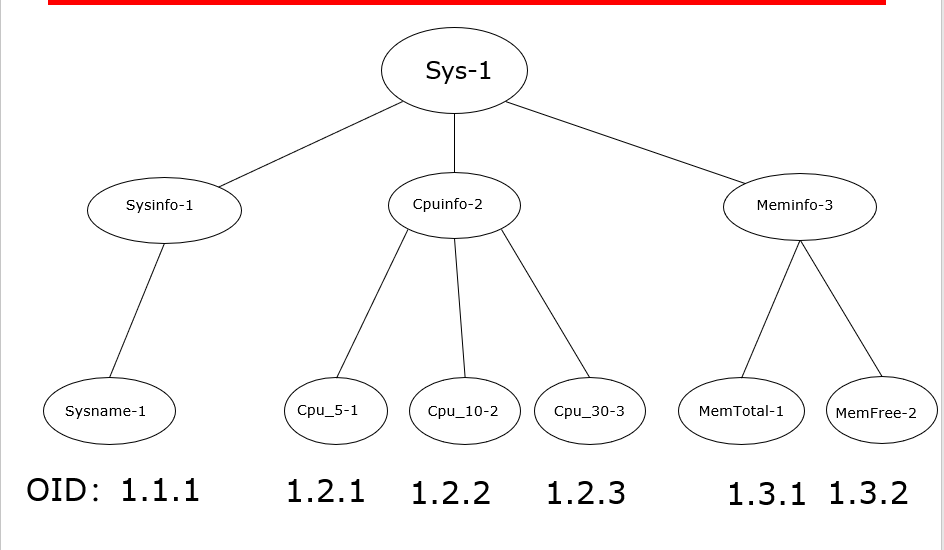
# 说明文档

## 应用实体目录树：



## 1.首先使用ASN.1描述SMI,基于SMI进一步编写应用实体的ASN.1说明文件，形成.txt文档

本身有适用于生产的ASN.1编译器，此次只是练习熟悉，不考虑生产应用

sln：SMI.txt + ASN1.txt 通用新更好

2.编写一个专用性ASN.1编译器，要求：

进行字符串匹配，直接获取关键字即可：

OBJECT IDENTIFIER 获取一行

OBJECT-TYPE 获取一行

SYNTAX获取一行 + 距离最近的 ::=一行

缺点：万一注释中有这个怎么办，无法处理

（1）做到边读文件边构建树形结构

树节点struct Node

{

char tag[10];/标签 名称

int id; //编号

char \*datatype; //数据类型

}

法一：jQuery ZTree Java JTree

法二：链表 不符合要求

（2）树形结构可以存储在内存中，也可以在.txt文件中

sln:通过构建三叉树，存储节点的标签 OID 数据类型

缺点：实现是在内存中，一旦节点过多，占用内存，可以放在文件中，用到时指定偏移量读取

（3）两端均使用OID标识节点（灵活点儿，不要写死在数组中）

sln:根据OID，去寻找对应节点的tag标签，返回字符串(OIDToTag函数)，然后通过字符串的判断，采集数据

## 3.Agent执行system（）函数 进行数据的采集

要求：使用头文件+小模块

sln:分别将sysinfo cpuinfo meminfo存储在data1 2 3.txt文件中

注意：不要输入一些不存在的OID，否则Agent 将直接退出

## 4.Manager分析显示数据

内存使用率：内存使用率要求保存，存在文件或者数据库中

系统信息：保存在文件中

CPU使用率：5 10 30seconds 使用率不要求保存，使用memCache作临时存储

说明：将TLV格式数据，转为key-value格式，主要存储的是value

sln:将接收到的数据按照空格拆分，取出Value字段，然后给一个key，存储到memcache中即可

## 5.数据传输编码BER

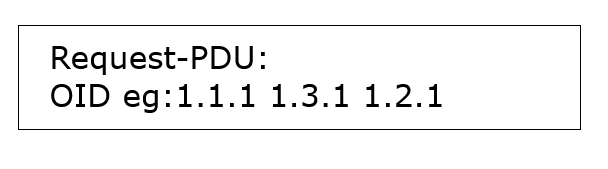
主要是char 类型编码，仿照UDP

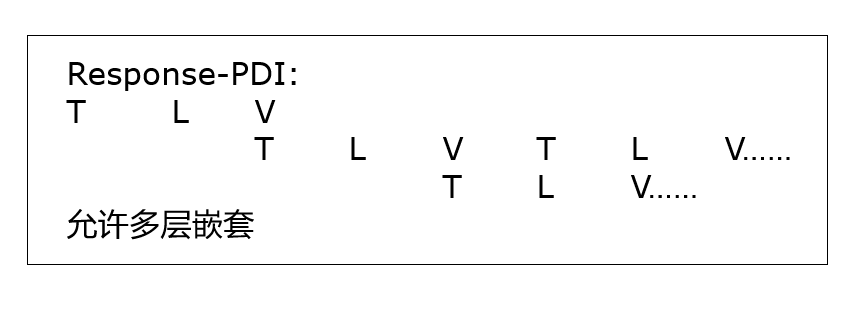
UDP：没有使用多进程 多线程处理，因为在此始终是单一Manager访问这个Agent 故而没有考虑那么多效率性能问题

首先，Manager发送请求报文，用于采集数据 request

然后，Manager收到响应报文，用于显示分析数据 analysis

PDU格式：





对char数据进行TLV编码

T:十进制表示 使用itoa,且无法使用unsigned char,由于string.h头文件内部很多函数，例如strcat strtok strcpy都是针对char\*类型编写的，编译会报错

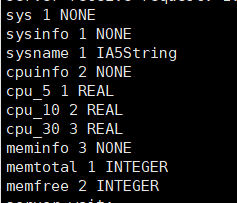
L:默认指的是后面发送char类型数据的长度，通过strlen()衡量，然后itoa,这里需要强调的是没有依据具体的类型去计数发送内容的大小，而是直接根据文件从读出数据的字节数，例如：cpu\_5 0.00 4B

V:默认指的是从文件中读取的数据内容

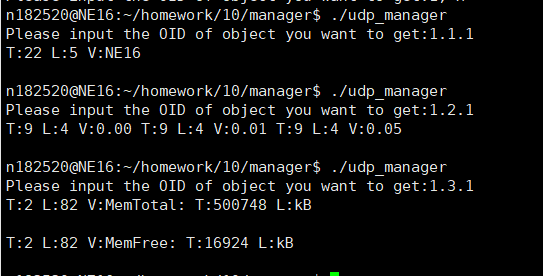
TLV集合转为char\*数据后，交由UDP数据包传输给请求方

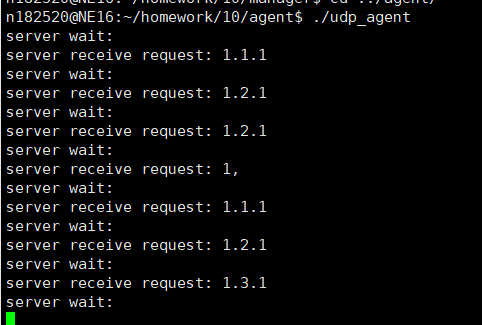
## 6.结果显示如下：

树结构：



Manager请求&Agent响应：





Memcache临时存储Cpu使用信息：

