第3章

ASN.1语言基础知识

ASN.1是一种对数据进行表示的语言标准，它被用于SNMP协议与MIB结构的定义。本章在讨论ASN.1基本概念的基础上，系统地介绍了ASN.1的数据类型、语法与编码规范，以及ASN.1在SNMP协议中的主要用途。

3.1 ASN.1的基本概念

抽象语法表示（Abstract Syntax Notation One, ASN.1）是一种独立于硬件结构的高级语言，提供对数据进行表示与编码的数据结构。ITU的X.680至X.683标准与ISO的8825-1至8825-4标准，定义ASN.1的语法、符号与参数等规范。ASN.1语法可以用来描述各种类型的数据，包括文本、图形、视频与音频等。图3-1给出了ASN.1的工作原理。ASN.1编码可以用来规范数据的传输过程，解决异构网络对数据理解上的二义性问题。ASN.1描述很容易映射成各种高级语言（如C、C++与Java等）的数据结构。

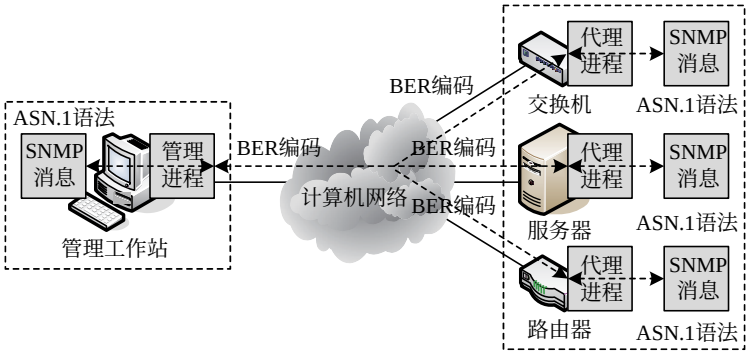

图3-1 ASN.1的工作原理

ASN.1标准支持以下几种编码规范：基本编码规范（Basic Encoding Rules, BER）、严格编码规范（Canonical Encoding Rules, CER）、唯一编码规范（Distinguished Encoding Rules, DER）、压缩编码规范（Packed Encoding Rules, PER）与XML编码规范（XML Encoding Rules, XER）。其中，BER是按类型、长度与值的顺序编码的规范；CER是对BER中不确定的编码加以严格定义的规范；DER是采用定长方式的严格编码规范；PER是减少数据类型的编码规范；XER是前几种ASN.1编码规范到XML的转换规则。

ASN.1最初是ITU针对电信协议设计而制定，后来成为被ISO组织接受的国际标准，并开

始广泛应用于各种网络协议的描述。ASN.1是用于定义应用程序数据的抽象语法，主要定义应用程序的数据结构与协议数据单元。目前，ASN.1主要应用于以下几种协议：SNMP（简单网络管理协议）、X.400（信息处理服务协议）、X.500（目录访问服务协议）、H.323（基于分组的多媒体通信VoIP结构）、RSA（公钥密码体制）、SET（安全电子商务）与Unicode（通用字符编码标准）。

3.2 ASN.1语法规范

3.2.1 ASN.1数据类型

1. 简单类型

简单类型是一种最基本的ASN.1数据类型。简单类型是指直接规定取值集合的类型，其中不会包括任何组件。简单类型相当于高级语言中的基本类型，例如C语言中的int、boolean等类型。简单类型是用于构造其他类型的基本单位。表3-1给出了ASN.1主要的简单类型。例如，INTEGER表示整数型，取值为正、负整数与0；BOOLEAN表示布尔型，取值为真（True）或假（False）。SMI定义SNMP协议中使用的简单类型，主要包括以下4种类型：INTEGER、OCTET STRING、NULL与OBJECT IDENTIFIER。

表3-1 ASN.1主要的简单类型

简 单 类 型	类 型 说 明
INTEGER	整数型（正、负整数与0的集合）
BOOLEAN	布尔型（True与False的集合）
REAL	实数型（正、负实数与0的集合）
ENUMERATED	枚举型（预定义的字符串与值的对应关系）
BIT STRING	比特流（二进制数组成的比特串）
OCTET STRING	字节流（二进制或十六进制数组成的字节串）
NULL	空类型（只有一个值null）
OBJECT IDENTIFIER	对象标识符（用于确定对象的一串整数）
OBJECT DESCRIPTION	对象描述符（用于确定对象的一串字符串）
EXTERNAL	自定义类型

2. 结构类型

结构类型也是一种基本的ASN.1数据类型。结构类型是指由多个组件构成的类型，每个组件是一个简单类型或结构类型。简单类型相当于高级语言中的结构类型，例如C语言中的struct类型。表3-2给出了ASN.1主要的结构类型。其中，SEQUENCE表示多个类型的有序集合，例如Student::==SEQUENCE{Name OCTET STRING, Age INTEGER}；SEQUENCE OF表示某个类型的有序集合，例如Student::==SEQUENCE OF Student。SMI定义SNMP协议中使用的结构类型，主要包括以下3种类型：SEQUENCE、SEQUENCE OF与CHOICE。

3. 标签类型

标签类型主要用于区分不同类型的数据，特别是SEQUENCE与SET中相同类型的组件。除了CHOICE与ANY类型之外，每种数据类型都拥有一个标签，由一个标签类与一个非负的标签值构成。标签值可以唯一地区分每种数据类型。ASN.1定义的标签类型主要有4种：通用类（Universal Class）、应用类（Application Class）、私有类（Private Class）与内容指定类

（Context-specific Class）。

表3-2 ASN.1主要的结构类型

结构类型	类型说明
SEQUENCE	多个类型的有序集合（类似于C语言中的结构）
SEQUENCE OF	某个类型的有序集合（类似于C语言中的数组）
SET	多个类型的无序集合（类似于SEQUENCE）
SET OF	某个类型的无序集合（类似于SEQUENCE OF）
CHOICE	多个类型的可选集合（取其中一个类型）
ANY	任意类型

通用类标签用于与应用无关的数据类型，每种ASN.1类型都有对应的标签值。例如，Student::=[UNIVERSAL 4]OCTET STRING。表3-3给出了ASN.1主要的通用类标签。应用类标签用于应用程序自定义的数据类型，可以由开发应用程序的企业或组织来定义。例如，Student::=[APPLICATION 1]SEQUENCE{Name OCTET STRING, Age INTEGER}。如果应用类标签与通用类标签的值相同，应用类标签将会代替相应的通用类标签。私有类标签用于企业或组织自定义的数据类型。

表3-3 ASN.1主要的通用类标签

通用类标签	类型说明
UNIVERSAL 1	布尔型（BOOLEAN）
UNIVERSAL 2	整数型（INTEGER）
UNIVERSAL 3	比特流（BIT STRING）
UNIVERSAL 4	字节流（OCTET STRING）
UNIVERSAL 5	空类型（NULL）
UNIVERSAL 6	对象标识符（OBJECT IDENTIFIER）
UNIVERSAL 7	对象描述符（OBJECT DESCRIPTION）
UNIVERSAL 8	自定义类型（EXTERNAL）
UNIVERSAL 9	实数型（REAL）
UNIVERSAL 10	枚举型（ENUMERATED）
UNIVERSAL 16	有序结构类型（SEQUENCE与SEQUENCE OF）
UNIVERSAL 17	无序结构类型（SET与SET OF）

内容指定类标签用于结构类型中的相同类型组件的区分，特别是在无序结构的SET类型中。例如，Student::==SET{Name[0] OCTET STRING, Identify[1] OCTET STRING, Age[2] INTEGER, Score[3] INTEGER}。在这个例子中，Student是由4个组件构成的无序类型，组件Name与Identify的类型是字节流，组件Age与Score的类型是整数型。这时，仅通过数据类型无法区分组件Name与Identify，同样也无法区分组件Age与Score，因此需要通过标签值来区别4个组件。

标签的定义方法可以分为两种：隐式（Implicit）与显式（Explicit）。其中，隐式标签通过改变组件类型的标签而生成，定义隐式标签的关键字是IMPLICIT；显式标签通过在组件类型的标签之外添加一个外部标签而生成，定义显式标签的关键字是EXPLICIT。如果没有将标签指定为隐式标签，ASN.1语法默认使用的是显式标签。隐式标签的类型可以看做与组件的类型相同；显式标签的类型可以看做有一个组件的结构类型。如果组件类型（如CHOICE或ANY）是不确定的，标签定义必须避免模糊不清。

3.2.2 ASN.1命名方法

ASN.1数据命名方法与高级编程语言（如C语言）相似。按照数据的用途与出现位置，ASN.1数据主要分为5类：关键字、类型名、模块名、宏名与对象名。其中，关键字是有专门用途的名称，全部字符大写（例如BEGIN）；类型名是数据类型的名称，首字符大写（例如AddressString）；模块名是模块的名称，首字符大写（例如AddressModule）；宏名是宏的名称，全部字符大写（例如ADDRESS）；对象名是数据对象的名称，首字符小写（例如ipAddress）。表3-4给出了ASN.1主要的关键字。

表3-4 ASN.1主要的关键字

关键字名称	关键字用途	关键字名称	关键字用途
DEFINITIONS	定义模块、类型或对象	EXPORTS	可被其他模块引用的数据类型
MACRO	定义宏	IDENTIFIER	零与正整数序列
BEGIN	定义模块或宏的开始	IMPLICIT	定义隐式标签
END	定义模块或宏的结束	EXPLICIT	定义显示标签
IMPORTS	从其他模块引用的数据类型		

ASN.1语句是由ASN.1数据与符号共同构成。每种ASN.1符号都有自己的用途，主要用于模块、数据类型、宏的定义，以及数据对象的赋值等操作。其中，“ ::= ”是最常见的ASN.1符号，其功能类似于C语言中的“ = ”，例如Age::=INTEGER；“ { } ”主要用于定义SEQUENCE、SET与ENUMERATED等类型的相关项，例如Student::==SEQUENCE {Name OCTET STRING, Age INTEGER}；“ -- ”用于表示注释部分，可以只出现在注释的开始部分表示一行，或出现在注释的开始与结束部分表示多行。表3-5给出了ASN.1主要的符号。

表3-5 ASN.1主要的符号

符号名称	符号用途	符号名称	符号用途
::=	模块、类型、宏定义或对象赋值	( )	子类型的开始与结束
{ }	相关项的开始与结束	..	子类型的取值范围
[ ]	可选项的开始与结束	--	注释部分的开始与结束
|	可选项中任选其一	...	多次重复的内容

3.2.3 ASN.1语法分析

ASN.1语法与高级编程语言（例如C语言）相似。下面将进行ASN.1语法分析。主要包括以下3部分内容：数据类型定义、宏定义与模块定义。这里将结合SNMP协议中的实例进行语法分析。

1. ASN.1数据类型定义

ASN.1数据类型是ASN.1语法中最基本的部分。数据类型定义主要包括4种：基本类型、子类型、SEQUENCE类型与SEQUENCE OF类型。其中，基本类型定义是对简单类型的定义；子类型定义是带约束条件的基本类型定义；SEQUENCE类型定义是对结构类型SEQUENCE的定义；SEQUENCE OF类型定义是对结构类型SEQUENCE OF的定义。下面给出的是ASN.1数据类型定义的格式：

```
<type name> ::= <type>  -- 基本类型定义
```

```
<type name> ::= <type>(<constraint>)  -- 子类型定义
<type name> ::= SEQUENCE {<type><name>,<type><name>,...}  -- SEQUENCE类型定义
<type name> ::= SEQUENCE OF <type> <type name>  -- SEQUENCE OF类型定义
```

ASN.1数据类型通常在ASN.1模块中定义。例如，在RFC1155-SMI模块的定义中，定义使用INTEGER类型的自定义类型Counter；在RFC1156-MIB模块的定义中，定义SEQUENCE结构的自定义类型ifEntry，使用SEQUENCE OF结构的自定义类型ifTable。下面给出的是几种数据类型的定义：

```
Counter ::= INTEGER  -- 基本类型定义
IpAddress ::= OCTET STRING
Counter ::= INTEGER(0..4294967295)  -- 子类型定义
IpAddress ::= OCTET STRING(SIZE(4))
ifEntry ::= SEQUENCE {  -- SEQUENCE类型定义
  ifIndex INTEGER, ifDescr OCTET STRING, ifType INTEGER, ifMtu INTEGER,
  ifSpeed Gauge, ifPhysAddress OCTET STRING, ifAdminStatus INTEGER,
  ifOperStatus INTEGER, ifLastChange TimeTicks, ifInOctets Counter,
  ifInUcastPkts Counter, ifInNUcastPkts Counter, ifInDiscards Counter,
  ifInErrors Counter, ifInUnknownProtos Counter, ifOutOctets Counter,
  ifOutUcastPkts Counter, ifOutNUcastPkts Counter, ifOutDiscards Counter,
  ifOutErrors Counter, ifOutQLen Gauge }
ifTable ::= SEQUENCE OF ifEntry  -- SEQUENCE OF类型定义
```

ASN.1数据对象赋值方法与高级编程语言（如C语言）相似。数据对象赋值语句的基本格式是：<object name><type>::=<value>。其中，object name是数据对象的名称，例如switchNumber、switchAddress与switchEntry；type是数据对象的数据类型，例如Counter、IpAddress与SEQUENCE；value是数据对象的赋值，例如24、192.168.1.1与{……}。下面给出的是几种数据类型的赋值：

```
switchNumber Counter ::= 24  -- 基本类型赋值
switchAddress IpAddress ::= 192.168.1.1
switchEntry ::= SEQUENCE {  -- SEQUENCE类型赋值
  ifIndex 24, ifDescr "ethernet-csmacd", ifType 6, ifMtu 1500, ifSpeed 10000000,
  ifPhysAddress "08-01-00-2A-10-C3", ifAdminStatus 1, ifOperStatus 1,
  ifLastChange 6000, ifInOctets 60000, ifInUcastPkts 1200, ifInNUcastPkts 600,
  ifInDiscards 10, ifInErrors 6, ifInUnknownProtos 0, ifOutOctets 120000,
  ifOutUcastPkts 2000, ifOutNUcastPkts 400, ifOutDiscards 6, ifOutErrors 2,
  ifOutQLen 20 }
```

2. ASN.1宏定义

ASN.1宏是ASN.1语法中的功能扩展部分，允许用户定义新的数据类型与数据赋值，以及一些起辅助作用的约束条件。用户可以在自己定义的ASN.1模块中，定义使用某种数据类型与相应的赋值关系的宏。ASN.1宏定义使用的关键字是MACRO，定义的内容位于BEGIN与END之间。其中，TYPE NOTATION定义数据类型使用的语法；VALUE NOTATION定义数据赋值使用的语法；辅助语法是可以选择的部分，用于说明数据类型的可选项。下面给出的是ASN.1宏定义的格式：

```
<macro name> MACRO ::=
```

```
BEGIN
  TYPE NOTATION ::= <syntax of new type>  -- 数据类型语法
  VALUE NOTATION ::= <syntax of new value>  -- 数据赋值语法
  <Assignment List>  -- 辅助语法
END
```

ASN.1宏通常在ASN.1模块中定义。例如，在RFC1155-SMI模块的定义中，定义用于创建管理对象的OBJECT-TYPE宏。数据类型语法包括3个部分：SYNTAX用于说明数据类型，例如INTEGER、OCTET STRING等；ACCESS用于说明访问模式，包括只读（read-only）、读写（read-write）、只写（write-only）与不可访问（not-accessible）；STATUS用于说明数据状态，包括强制（mandatory）、可选（optional）与废弃（obsolete）。数据赋值语法用于说明赋值方式。下面给出的是OBJECT-TYPE宏的定义：

```
OBJECT-TYPE MACRO ::=
BEGIN
  TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                    "ACCESS" Access
                    "STATUS" Status
  VALUE NOTATION ::= value (VALUE ObjectName)
  Access ::= "read-only" | "read-write" | "write-only" | "not-accessible"
  Status ::= "mandatory" | "optional" | "obsolete"
END
```

ASN.1宏通常在ASN.1模块中使用。例如，在RFC1156-MIB模块的定义中，定义使用OBJECT-TYPE宏的管理对象sysDescr。其中，数据类型是字节流（OCTET STRING）；访问模式是只读（read-only）；数据状态是强制（mandatory）；赋值方式是system组的第一个对象。下面给出的是OBJECT-TYPE宏的使用：

```
sysDescr OBJECT-TYPE
  SYNTAX  OCTET STRING
  ACCESS  read-only
  STATUS  mandatory
::= { system 1 }
```

3. ASN.1模块定义

ASN.1模块是ASN.1语法中的基本构造块，通常用于定义一类相关的数据结构。模块在规模方面没有任何限制，一个软件开发商可以定义一个模块，一款软件产品可以定义一个模块，软件中的一个功能也可以定义一个模块。在SNMP协议的描述中，SMI结构、SNMP协议与MIB库分别定义在不同模块中。模块定义使用的关键字是DEFINITIONS，定义的内容位于BEGIN与END之间。其中，EXPORTS是可供其他模块引用的定义；IMPORTS是从其他模块引用的定义。下面给出的是ASN.1模块的定义：

```
<module name> DEFINITIONS ::=
BEGIN
  EXPORTS  -- 本模块中可供其他模块引用的定义
  IMPORTS  -- 本模块使用从其他模块引用的定义
  Assignment List  -- 宏定义、类型定义与对象赋值
END
```

RFC1155-SMI模块定义SNMP协议使用的SMI，主要是可供RFC1156-MIB与RFC1157-SNMP模块使用的数据结构。EXPORTS是可供其他模块引用的定义，包括internet、OBJECT-TYPE与NetworkAddress。这些定义可以分为3种用途：SNMP涉及的MIT结点，例如internet、directory与mgmt等；可供SNMP与MIB使用的宏，例如OBJECT-TYPE；可供SNMP与MIB使用的数据类型，例如NetworkAddress、IpAddress与Counter等。下面给出的是RFC1155-SMI模块的定义：

```
RFC1155-SMI DEFINITIONS ::=
BEGIN
  EXPORTS
    internet, directory, mgmt, experimental, private, enterprises, OBJECT-TYPE,
    ObjectName, ObjectSyntax, SimpleSyntax, ApplicationSyntax, NetworkAddress,
    IpAddress, Counter, Gauge, TimeTicks, Opaque;
  -- the path to the root
  internet OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
  directory OBJECT IDENTIFIER ::= { internet 1 }
  mgmt OBJECT IDENTIFIER ::= { internet 2 }
  experimental OBJECT IDENTIFIER ::= { internet 3 }
  private OBJECT IDENTIFIER ::= { internet 4 }
  enterprises OBJECT IDENTIFIER ::= { private 1 }
  -- definition of object types
  OBJECT-TYPE MACRO ::=
  BEGIN
    TYPE NOTATION ::= "SYNTAX" type (TYPE ObjectSyntax)
                      "ACCESS" Access
                      "STATUS" Status
    VALUE NOTATION ::= value (VALUE ObjectName)
    Access ::= "read-only" | "read-write" | "write-only" | "not-accessible"
    Status ::= "mandatory" | "optional" | "obsolete"
  END
  -- names of objects in the MIB
  ObjectName ::= OBJECT IDENTIFIER
  -- syntax of objects in the MIB
  ObjectSyntax ::= CHOICE { simple SimpleSyntax,
                            application-wide ApplicationSyntax }
  SimpleSyntax ::= CHOICE { number INTEGER, string OCTET STRING,
                            Object OBJECT IDENTIFIER, empty NULL }
  ApplicationSyntax ::= CHOICE { address NetworkAddress, counter Counter,
                                 gauge Gauge, ticks TimeTicks, arbitrary Opaque }
  -- application-wide types
  NetworkAddress ::= CHOICE { internet IpAddress }
  IpAddress ::= [APPLICATION 0] IMPLICIT OCTET STRING (SIZE (4))
  Counter ::= [APPLICATION 1] IMPLICIT INTEGER (0..4294967295)
  Gauge ::= [APPLICATION 2] IMPLICIT INTEGER (0..4294967295)
  TimeTicks ::= [APPLICATION 3] IMPLICIT INTEGER (0..4294967295)
  Opaque ::= [APPLICATION 4] IMPLICIT OCTET STRING
END
```

RFC1156-MIB模块定义SNMP协议使用的MIB，主要是可供RFC1157-SNMP模块使用的具体的管理对象。RFC1156-MIB模块定义MIB中的8个组，例如system、interfaces与ip与snmp等；定义构成每个组的管理对象，例如system组中的sysDescr、sysObjectID与sysUpTime等。RFC1156-MIB模块定义MIB中的管理对象时，使用由RFC1155-SMI模块定义的数据类型，例如mgmt、OBJECT-TYPE、NetworkAddress与IpAddress等。下面给出的是RFC1156-MIB模块的定义：

```
RFC1156-MIB DEFINITIONS ::=
BEGIN
  IMPORTS
    mgmt, OBJECT-TYPE, NetworkAddress, IpAddress, Counter, Gauge, TimeTicks
    FROM RFC1155-SMI;
  mib OBJECT IDENTIFIER ::= { mgmt 1 }
  system OBJECT IDENTIFIER ::= { mib 1 }
  interfaces OBJECT IDENTIFIER ::= { mib 2 }
  at OBJECT IDENTIFIER ::= { mib 3 }
  ip OBJECT IDENTIFIER ::= { mib 4 }
  icmp OBJECT IDENTIFIER ::= { mib 5 }
  tcp OBJECT IDENTIFIER ::= { mib 6 }
  udp OBJECT IDENTIFIER ::= { mib 7 }
  egp OBJECT IDENTIFIER ::= { mib 8 }
  -- the System group
  sysDescr OBJECT-TYPE
    SYNTAX  OCTET STRING
    ACCESS  read-only
    STATUS  mandatory
  ::= { system 1 }
  sysObjectID OBJECT-TYPE
    SYNTAX  OBJECT IDENTIFIER
    ACCESS  read-only
    STATUS  mandatory
  ::= { system 2 }
  sysUpTime OBJECT-TYPE
    SYNTAX  TimeTicks
    ACCESS  read-only
    STATUS  mandatory
  ::= { system 3 }
  -- the Interfaces group
  ifNumber OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
  ::= { interfaces 1 }
  -- the Interfaces table
  ifTable OBJECT-TYPE
    SYNTAX  SEQUENCE OF ifEntry
    ACCESS  read-write
    STATUS  mandatory
```

```
  ::= { interfaces 2 }
  ifEntry OBJECT-TYPE
    SYNTAX  ifEntry
    ACCESS  read-write
    STATUS  mandatory
  ::= { ifTable 1 }
  ifEntry ::= SEQUENCE {
    ifIndex INTEGER, ifDescr OCTET STRING, ifType INTEGER, ifMtu INTEGER,
    ifSpeed Gauge, ifPhysAddress OCTET STRING, ifAdminStatus INTEGER,
    ifOperStatus INTEGER, ifLastChange TimeTicks, ifInOctets Counter,
    ifInUcastPkts Counter, ifInNUcastPkts Counter, ifInDiscards Counter,
    ifInErrors Counter, ifInUnknownProtos Counter, ifOutOctets Counter,
    ifOutUcastPkts Counter, ifOutNUcastPkts Counter, ifOutDiscards Counter,
    ifOutErrors Counter, ifOutQLen Gauge }
  ifIndex OBJECT-TYPE
    SYNTAX INTEGER
    ACCESS read-only
    STATUS mandatory
  ::= { ifEntry 1 }
  ifDescr OBJECT-TYPE
    SYNTAX OCTET STRING
    ACCESS read-only
    STATUS mandatory
  ::= { ifEntry 2 }
  省略
END
```

RFC1157-SNMP模块定义SNMP协议的基本内容，主要包括SNMP消息与协议数据单元。RFC1157-SNMP模块定义SNMP消息包括3个部分：版本号（Version）、团体（Community）与协议数据单元（PDU）。其中，PDU主要包括5种类型：GetRequest-PDU、GetNextRequest-PDU、SetRequest-PDU、GetResponse-PDU、Trap-PDU。RFC1157-SNMP模块定义SNMP协议内容时，使用由RFC1155-SMI模块定义的数据类型，例如NetworkAddress、IpAddress与TimeTicks等。下面给出的是RFC1157-SNMP模块的定义：

```
RFC1157-SNMP DEFINITIONS ::=
BEGIN
  IMPORTS
    ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
    FROM RFC1155-SMI;
  -- top-level message
  Message ::= SEQUENCE {
    version INTEGER { version-1(0) }, community OCTET STRING, data ANY }
  -- protocol data units
  PDUs ::= CHOICE {
    get-request GetRequest-PDU, get-next-request GetNextRequest-PDU, get-
    response GetResponse-PDU, set-request SetRequest-PDU, trap Trap-PDU }
  -- PDUs
  GetRequest-PDU ::= [0] IMPLICIT PDU
```

```
  GetNextRequest-PDU ::= [1] IMPLICIT PDU
  GetResponse-PDU ::= [2] IMPLICIT PDU
  SetRequest-PDU ::= [3] IMPLICIT PDU
  PDU ::= SEQUENCE {
    request-id INTEGER, error-status INTEGER { noError(0),tooBig(1),
    noSuchName(2), badValue(3),readOnly(4),genErr(5) }, error-index INTEGER,
    variable-bindings VarBindList }
  Trap-PDU ::= [4] IMPLICIT SEQUENCE {
    enterprise OBJECT IDENTIFIER, agent-addr NetworkAddress, generic-trap INTEGER
    { coldStart(0),warmStart(1),linkDown(2),linkUp(3),authenticationFailure(4),
    egpNeighborLoss(5),enterpriseSpecific(6) }, specific-trap INTEGER, time-stamp
    TimeTicks, variable-bindings VarBindList }
  -- variable bindings
  VarBind ::= SEQUENCE { Name ObjectName, value ObjectSyntax }
  VarBindList ::= SEQUENCE OF VarBind
END
```

3.3 BER的基本概念

BER是ASN.1语言支持的主要编码规范之一。ASN.1语法用于完成数据的表示，定义网络协议的操作类型与数据结构；BER编码用于完成数据的编码，避免数据在异构网络中传输时出现二义性。ITU的X.690标准与ISO的8825—1标准，定义包括BER、CER与DER的几种编码规范。在基于SNMP的网络管理应用中，发送方通过ASN.1语法构造SNMP消息，然后将SNMP消息按BER规则进行编码，并将BER编码通过网络发送给接收方，最后接收方对BER编码进行解码获得SNMP消息。

BER编码方法主要分为3种类型：简单定长、结构化定长与结构化非定长。其中，简单定长方法适用于除字节流之外的简单类型，以及对简单类型通过隐式标签生成的类型，该方法要求预先知道数值的长度；结构化定长方法适用于字节流的简单类型、结构类型与在二者基础上通过隐式标签生成的类型，以及对任何类型通过显式标签生成的类型，该方法要求预先知道数值的长度；结构化非定长方法适用于结构化定长方法生成的类型，但是该方法不要求预先知道数值的长度。

BER编码方法采用TLV（Type Length Value）结构，每个字节按照类型（Type）、长度（Length）与数值（Value）的顺序编码。图3-2给出了BER编码的TLV结构。类型字节的第7与第8位表示标签类型，例如数值00表示通用类（Universal Class），数值01表示应用类（Application Class）；第6位表示简单类型或结构化类型，其中数值0表示简单类型，数值1表示结构化类型；第1至第5位表示标签号，对于简单类型是通用类的标签号，对于其他类型是定义式的标签号。表3-6给出了标签类型的数值定义。

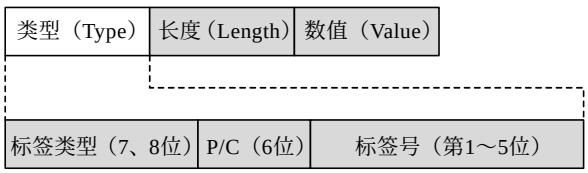

图3-2 BER编码的TLV结构

BER编码首先生成表示数据类型的字节，然后生成表示数据长度的字节，最后按长度依次生成表示数值的字节。图3-3给出了一个BER编码的例子。对于数值为1的整数，类型字节第7、8位为通用类（取值00），第6位为简单类型（取值0），第1至5位为INTEGER的标签号（取值00010），后面2个字节分别为数据长度与数值；对于数值为J0HN的字节流，类型字节第7、8位为通用类（取值00），第6位为简单类型（取值0），第1至5位为OCTET STRING的标签号（取值00100），后面5个字节分别为数据长度与数值。

表3-6 标签类型的数值定义

标 签 类 型	第8位	第7位
通用类	0	0
应用类	0	1
私有类	1	0
面向内容类	1	1

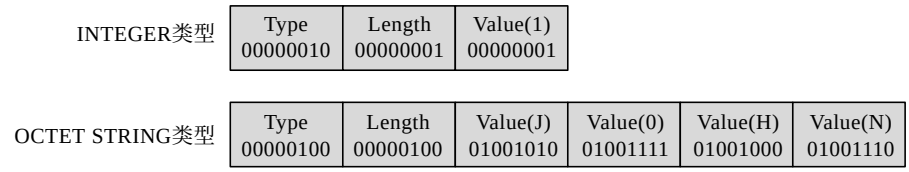

图3-3 一个BER编码的例子

3.4 本章总结

本章主要讲述了以下内容：

- 抽象语法表示（ASN.1）是一种独立于硬件结构的高级语言，提供对数据进行表示与编码的数据结构。ASN.1可以用来描述各种类型的数据，包括文本、图形、视频与音频等。
- ASN.1数据类型主要包括3类：简单类型、结构类型与标签类型。简单类型是直接规定取值集合的类型，其中不会包括任何组件；结构类型是由多个组件构成的类型，每个组件是一个简单类型或结构类型；标签类型用于区分不同类型的数据，特别是SEQUENCE与SET中相同类型的组件。
- ASN.1数据主要分为5类：关键字、类型名、模块名、宏名与对象名。其中，关键字是有专门用途的名称，全部字符大写；类型名是数据类型的名称，首字符大写；模块名是模块的名称，首字符大写；宏名是宏的名称，全部字符大写；对象名是数据对象的名称，首字符小写。
- BER是ASN.1支持的主要编码规范之一。BER编码用于完成数据的编码，避免数据在异构网络中传输时出现二义性。BER编码方法采用TLV结构，每个字节按照类型（Type）、长度（Length）与数值（Value）的顺序编码。

3.5 本章习题

1. 单项选择题

（1）在ASN.1编码规范中，基本编码规范的英文缩写是_____。

A. BER　　B. CER　　C. DER　　D. PER

（2）关于ASN.1语言的描述中，正确的是_____。

A. ASN.1是一种基于特定硬件的高级语言

B. ASN.1只能描述音频与视频类型的数据

C. ASN.1最初是IEEE针对电信协议而设计的

D. ASN.1用于简单网络管理协议的描述

（3）在以下几种数据类型中，不属于简单类型的是_____。

A. BOOLEAN　　B. INTEGER　　C. SEQUENCE　　D. ENUMERATED

（4）关于标签类型的描述中，错误的是_____。

A. 标签类型主要用于区分不同类型的数据

B. 标签类型只包括通用类、应用类与私有类

C. 通用类标签用于与应用无关的数据类型

D. 私有类标签用于企业自定义的数据类型

（5）在以下几种关键字中，用于定义隐式标签的是_____。

A. EXPORTS　　B. IMPORTS　　C. EXPLICIT　　D. IMPLICIT

（6）关于ASN.1命名方法的描述中，正确的是_____。

A. 关键字命名需要首字符小写　　B. 类型名命名需要首字符小写

C. 对象名命名需要首字符小写　　D. 模块名命名需要首字符小写

（7）在ASN.1结构类型中，表示多个类型的有序集合的是_____。

A. SEQUENCE　　B. CHOICE　　C. BIT STRING　　D. OCTET STRING

（8）关于BER编码的描述中，错误的是_____。

A. BER编码用于ASN.1数据的编码与解码操作

B. BER编码按照类型、长度与数值的顺序编码

C. BER编码只支持简单定长与结构化定长方式

D. BER编码避免数据在异构网络传输的二义性

（9）在通用型标签中，标签值2表示的数据类型是_____。

A. 实数型　　B. 整数型　　C. 布尔型　　D. 枚举型

2. 填空题

（1）在SNMP协议中，协议结构与数据类型由_____定义。

（2）在ASN.1简单类型中，表示字节流类型的关键字是_____。

（3）在ASN.1语法中，表示对象赋值的符号是_____。

（4）在ASN.1编码规范中，采用定长方式严格编码规范的是_____。

（5）在ASN.1数据类型中，表示某个类型有序集合的关键字是_____。

（6）在ASN.1语法中，定义模块或宏的开始的关键字是_____。

（7）BER编码方法的每个字节按类型、_____与数值的顺序编码。

（8）在ASN.1语法中，类型名的基本命名规则为首字符_____。

（9）如果某个数据进行BER编码后为000000100000000100000110，则该数据进行解码后的内容是_____。

3. 问答题

（1）ASN.1数据主要分为几种类型？每种类型的基本特点是什么？

（2）ASN.1标签主要分为几种类型？每种标签的基本特点是什么？

（3）ASN.1与BER的区别与联系是什么？请简要说明。

（4）ASN.1数据的命名规则是什么？请简要说明。

（5）BER编码的基本规则是什么？请简要说明。