UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANA

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

ESPECIALIZAÇÃO EM TELEINFORMATICA E REDES DE COMPUTADORES

CRISTIANO FERREIRA SIERRA

DESENVOLVIMENTO DE UM PROGRAMA AGENTE SNMP PARA SIMULAR O FUNCIONAMENTO DE UMA MIB

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA

2011

CRISTIANO FERREIRA SIERRA

DESENVOLVIMENTO DE UM PROGRAMA AGENTE SNMP PARA SIMULAR O FUNCIONAMENTO DE UMA MIB

Monografia apresentada como requisito parcial para a obtenção do titulo de Especialista do Curso de Teleinformática e Redes de Computadores, Departamento de Eletrônica, Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Armando Rech Filho

CURITIBA

2011

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação do *Campus* Curitiba
Departamento Acadêmico de Eletrônica
Especialização em Teleinformática e Redes de Computadores

TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UM PROGRAMA AGENTE SNMP PARA SIMULAR O FUNCIONAMENTO DE UMA MIB

por

CRISTIANO FERREIRA SIERRA

Esta monografia foi apresentada às 14:30h do dia 07 de MARÇO de 2012 como requisito parcial para a obtenção do título de ESPECIALISTA EM TELEINFORMÁTICA E REDES DE COMPUTADORES, Universidade Tecnológica Federal do Paraná. O candidato foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com a nota 9,5 (NOVE INTEIROS E CINCO DÉCIMOS)

Prof. Dr. Armando Bech Filho
(UTFPR)

Prof. Dr. Walter Godoy Júnior
(UTFPR)

Visto da Coordenação

Prof. Dr. Walter Godoy Júnior
Coordenador do Curso

AGRADECIMENTOS

Como em tudo o que se faz na vida, este trabalho também contou com o apoio, carinho e colaboração de muitas pessoas, sem as quais não teria sido possível acontecer.

Inicialmente agradeço aos familiares, amigos e colegas que se mantiveram do nosso lado apoiando e incentivando a permanecermos confiantes em nossos objetivos.

Ao Prof. Dr. Armando Rech Filho, pelo apoio com o qual me acolheu e me acompanhou e finalmente a Deus.

RESUMO

Sierra, Cristiano Ferreira. Desenvolvimento de um programa agente SNMP para simular o funcionamento de uma MIB. 2012. 40 f. Monografia (Especialização em Teleinformática e Redes de Computadores) – Departamento Acadêmico de Eletrônica da Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

Este trabalho apresenta o desenvolvimento de um agente de gerência SNMP que possua a capacidade de simular o funcionamento de uma MIB qualquer, bem como possa utilizar dados de simulação fornecidos pelo usuário. Discutem-se os conceitos do protocolo SNMP, a criação e estruturação dos dados da MIB e os programas utilizados para o desenvolvimento de um agente SNMP. Como resultado tem-se um aplicativo agente SNMP simulador de MIB.

Palavras chave: Agente. Snmp. Mib. Java. Gerência.

ABSTRACT

Sierra, Cristiano Ferreira. Development of an SNMP agent to simulate the operation of a MIB. 2012. 41 f. Monografia (Especialização em Teleinformática e Redes de Computadores) – Departamento Acadêmico de Eletrônica da Universidade Tecnológica Federal do Paraná. Curitiba, 2011.

It presents the development of an SNMP management agent that has the ability to simulate the operation of any MIB, and can use simulation data supplied by the user. It discusses the concepts of the SNMP protocol, the creation and structuring of data for the MIB and the programs used for development of an SNMP agent. As a result an application MIB SNMP agent simulator is deployed.

Palavras chave: Agent. Snmp. Mib. Java. Management.

LISTA DE FIGURAS

LISTA DE ACRÔNIMOS

ASN.1	Abstract Syntax Notation One
CIM	Common Information Model
CIM-XML	Common Information Model -Extensible Markup Language
DMTF	Distributed Management Task Force
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
JDMK	Java Dynamic Management Kit
JMX	Java Management Extensions
OID	Object Identifier
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
RFC	Request for Comments
RMON	Remote Network Monitoring
SGMP	Simple Gateway Monitoring Protocol
SLP	Service Location Protocol
SMI	Structure of Management Information
SONET	Synchronous Optical Networking
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
USM	User-Based Security Model
VACM	View-based Access Control Model
WBEM	Web-Based Enterprise Management
WMI	Windows Management Instrumentation

SUMÁRIO

1 INTRODUÇÃO

Atualmente o processo de gerenciamento de dispositivos e sistemas já não é mais um opcional para as empresas, mas se tornou obrigatório para toda empresa que busca crescimento utilizando dados concretos e reais do seu negócio, uma vez que suas redes de comunicação precisam ser confiáveis, ter alta disponibilidade e desempenho condizente com as demandas.

Ou seja, com o avanço dos computadores e da fácil conectividade entre eles, obter dados de qualquer parte do planeta hoje em dia é uma questão de segundos. E a empresa que não souber utilizar esta vantagem estará fadada ao atraso tecnológico, e consequentemente a se posicionar atrás de seus concorrentes.

Buscando atender a esta necessidade de mercado com relação à gerência que cresce a cada dia, as instituições de ensino que fornecem cursos de tecnologia atualmente oferecem conteúdos programáticos relativos ao gerenciamento de redes e sistemas, e o protocolo de gerência mais trabalhado é o *Simple Network Management Protocol* (SNMP) criado e mantido pelo *Institute of Electrical and Electronic Engineers* (IEEE), que pode ser utilizado em qualquer elemento conectado a uma rede, desde um dispositivo qualquer até um sistema aplicativo de processo de negócio de uma organização.

O protocolo SNMP é utilizado na conversação entre um agente e um gerente. O agente é um programa que captura, guarda e caso seja necessário transmite informações do dispositivo ou sistema a ser monitorado para o gerente; e o gerente é o programa que busca estas informações no agente e as disponibiliza de forma inteligível para os usuários, utilizando tabelas e gráficos.

Todos os dados utilizados pelo protocolo são formatados conforme um modelo de informação chamado de *Management Information Base* (MIB). E este modelo deve seguir um padrão defino pelo IEEE. A construção de uma MIB é muito discutida nas instituições de ensino, pois dela nasce todo o conceito de como trabalha o gerenciamento SNMP, e é ela quem vai determinar quais objetos podem ou não ser gerenciados em um elemento de rede.

O processo de construção de uma MIB deveria ser acompanhado do desenvolvimento de um agente para demonstrar o seu funcionamento, o que não é uma tarefa trivial e exige dos estudantes, além dos protocolos, também bons conhecimentos de programação.

Justifica-se este trabalho pela colaboração que pode trazer para um entendimento mais profundo das capacidades e funcionalidades do gerenciamento utilizando o protocolo SNMP e seus componentes, gerente, agente, operações de gerência e a própria MIB.

O objetivo deste trabalho é construir um programa agente SNMP genérico para realizar simulação do funcionamento de uma MIB, que seja capaz de entender a estrutura de qualquer MIB e gerar valores para seus objetos, a serem respondidos quando demandados por um programa gerente.

São objetivos específicos:

Abordar a atual situação dos protocolos de gerência de redes, bem como suas funções principais e características, mostrando o protocolo SNMP;

Descrever a MIB bem como sua estrutura de dados e as operações de gerência;

Identificar os programas de apoio à construção de agentes e gerentes que podem ser utilizados no trabalho, justificando a escolha;

Elaborar o programa proposto, documentando a sua lógica e forma de utilização;

Apresentar os resultados dos testes realizados;

A metodologia de pesquisa empregada para o desenvolvimento do trabalho pode ser classificada como exploratória, pois busca compreender o processo de desenvolvimento de agentes e simulação de funcionamento de MIBs construídas para o gerenciamento de recursos em redes de comunicação de dados.

Ela compreende o tratamento de fontes secundárias, através da revisão bibliográfica abordando conceitos e informações existentes na literatura, incluindo livros, artigos, trabalhos acadêmicos e científicos dentre outros.

A partir dos conhecimentos adquiridos é realizado um estudo de caso que compreende o desenvolvimento e os testes de um programa agente SNMP.

Para tanto o trabalho está assim estruturado: no capítulo 2 são apresentados os estudos referentes à gerência de redes e seus protocolos; o capítulo 3 aborda a criação da MIB e sua estrutura, e as operações de gerência; no capítulo 4 são analisados conceitos para o desenvolvimento de uma agente SNMP, bem como a descrição do programa criado e são mostrados os resultados dos testes realizados com o programa desenvolvido.

2 GERÊNCIA DE REDES E PROTOCOLOS

Segundo Saydam e Magedanz (1996),

> o gerenciamento de rede inclui a disponibilização, a integração e a coordenação de elementos de hardware, software e humanos, para monitorar, testar, consultar, configurar, analisar, avaliar e controlar os recursos da rede, e de elementos, para satisfazer às exigências operacionais, de desempenho e de qualidade de serviço em tempo real a um custo razoável.

Observando bem esta breve e concisa descrição sobre gerência de rede, podem-se perceber três componentes que existirão em todos os métodos de gerenciamento. Os componentes são: o gerente, os dispositivos gerenciados e o protocolo de conversa entre ambos.

O gerente normalmente é ponto da arquitetura onde o ser humano atua; esta aplicação é a central da gerência de rede, é ela quem controla a aquisição de dados, o processamento e análise conforme prévia configuração, e também é ela quem formata os dados para apresentação como gráficos e alarmes. A partir dos dados disponibilizados por esta ferramenta é que o administrador humano entra em ação, podendo controlar e interagir com os dispositivos gerenciados.

O dispositivo gerenciado pode ser qualquer elemento que consiga se conectar e comunicar pela rede gerenciada. Ele pode ser um computador, um roteador, uma impressora, um celular, uma catraca, um sensor, um banco de dados, uma aplicação, dentre outros elementos. Para que cada dispositivo possa ser gerenciado ele deve conter objetos ou classes de dados a serem gerenciados, como quantidade de portas de um roteador e suas respectivas velocidades ou quantas páginas um usuário imprimiu. Quando estes objetos estão sendo gerenciados as suas informações são coletadas e armazenadas dentro de uma MIB, isto ocorre devido a uma aplicação chamada de agente de gerenciamento que existe em todo dispositivo gerenciado. É esta aplicação que organiza e disponibiliza os dados ao gerente.

E por final o protocolo de conversa entre ambos, ou seja, a língua e método em que as partes da rede gerenciada irão se comunicar. Ele é chamado de protocolo de gerenciamento de rede, onde todos os dispositivos que serão

gerenciados devem utilizar o mesmo protocolo. Vale ressaltar que o protocolo de gerenciamento em si não gerência a rede, mas sim provê uma interface de comunicação entre gerente e gerenciado.

Dentre os protocolos de gerência mais conhecidos e utilizados no mercado estão o SNMP, utilizado neste trabalho e o *Web-Based Enterprise Management* (WBEM). Ambos partilham da mesma ideia central, em seguida eles são apresentados de forma resumida. (KUROSE; ROSS, 2003)

2.1 O SNMP

O SNMP ao contrário do que muitos pensam não é apenas um protocolo, mas sim um conjunto de especificações sobre o gerenciamento de redes que inclui o protocolo SNMP em suas especificações (STALLINGS, 2005). Ele foi desenvolvido e disponibilizado na década de 80, e isto foi quando as discussões sobre gerenciamento de redes começaram a fermentar por todos os lados. Por ter sido disponibilizado rapidamente e por ser extremamente simples o SNMP foi logo adotado em larga escala, sendo que hoje ele é o modelo de gerenciamento mais utilizado (KUROSE; ROSS, 2003).

Desde seu lançamento o SNMP teve três versões principais, a primeira foi o SNMPv1.

As raízes da estrutura do SNMPv1 foram baseadas no protocolo *Simple Gateway Monitoring Protocol* (SGMP). O mesmo time que havia desenvolvido o SGMP desenvolveu o SNMPv1 em seguida utilizando os mesmos princípios, o que possibilitou um desenvolvimento rápido em poucos meses. (KUROSE; ROSS, 2003)

O SNMPv1 abrange todos os pontos descritos no inicio desse capitulo. O banco de dados elaborado para o protocolo foi chamado de MIB, que é a base de informações gerenciadas. Para a troca de informações entre gerentes e gerenciados foi utilizado do protocolo *User Datagram Protocol* (UDP) que não é orientado a conexão.

Foram criadas três tipos de mensagens de comunicação: *GetRequest*, *GetNextRequest* e *SetRequest*. As duas primeiras são variações da função *get* (coletar dados) e a terceira serve para modificar informações na MIB. Existe ainda

uma função chamada *trap*, que é uma resposta a um evento ocorrido no dispositivo gerenciado que envia uma mensagem para o gerente.

O SNMPv2 contem todas as diretivas da versão anterior, porém em seu desenvolvimento buscou-se melhorar o desempenho em varias áreas que eram deficientes na versão 1, como a desempenho do protocolo, a segurança, a confidencialidade e a comunicação entre gerentes, para prover uma arquitetura de gerenciamento distribuído.

Uma das principais diferenças com seu antecessor foi o desenvolvimento de uma função get recursiva, cujo nome é *Get-BulkRequest*, já que com esta função é possível obter grandes quantidades de dados mais facilmente e com uma simples chamada de função. Também foi incrementada a tabela de tipos de dados permitidos, com várias novidades como os tipos *Counter*, *Gauge*, *TimeTicks* e outros tipos de *string*.

Ainda que em sua segunda versão tenha se aprimorado bastante o protocolo, a segurança do protocolo ainda continuava falha, por ter retornado à mesma da versão 1, já que seu mecanismo foi considerado muito complexo para a época. Partindo desta deficiência os desenvolvedores recriaram toda a parte relativa à segurança do protocolo e lançaram por volta do ano 2002 uma nova versão do protocolo, a versão três. Esta versão engloba as *Request for Comments* (RFC) 3411 a 3415, as quais descrevem todas as novidades do protocolo e seus novos métodos de segurança e criptografia.

Dentre as melhorias feitas nesta versão as mais importantes são relativas à autenticação, privacidade e controle de acesso. Sendo que os dois primeiros são relativos ao User-Based Security Model (USM) e o terceiro é definido no View-Based Access Control Model (VACM). O USM tem dois importantes papéis, garante a troca de mensagens do protocolo, ou seja, garante o conteúdo das informações e a autenticidade do seu emissor e também aplica a criptografia nos dados transmitidos. Já o VACM serve para adicionar níveis de acesso a MIB, o que possibilita uma gerência de dados com níveis de acesso nos agentes (STALLINGS, 2005).

2.2 O WBEM

O WBEM é um conjunto de padrões que abordam o gerenciamento, e é mantido pelo *Distributed Management Task Force* (DMTF). O padrão nasceu da necessidade de se trocar informações entre arquiteturas e tecnologias diferentes, e tendo nascido muito tempo depois do SNMP, o WBEM tem características e formatos modernos comparados aos utilizados pelo SNMP (DMTF, 2012).

Dentre os padrões utilizados pelo WBEM estão o *Common Information Model* (CIM), CIM-XML, CIM *Query Language*, WBEM *Discovery using Service Location Protocol* (SLP) e o WBEM *Universal Resource Identifier* (URI) *mapping* (DMTF, 2012). Todos estes padrões e outros menores visam ser muito extensíveis e plugáveis, para que possam ser utilizados em qualquer dispositivo ou serviço.

Tal qual o SNMP, ele é um padrão aberto onde qualquer empresa pode utiliza-lo, e um exemplo de sua utilização é a empresa Microsoft que o emprega na ferramenta *Windows Management Instrumentation* (WMI). Os três principais componentes do padrão são: o CIM que mantem a base de informações gerenciáveis assim como a MIB do SNMP, o CIM-XML que é o método de codificação das informações e o *Hypertext Transfer Protocol* (HTTP) para o transporte e acesso das informações.

3 A MIB E AS OPERAÇÕES DE GERÊNCIA

A base de informações de gerenciamento é um como banco de dados virtual que guarda em seus objetos os dados coletados da rede. As informações podem ser definidas pelo próprio dispositivo gerenciado ou pelo gerente SNMP. (KUROSE; ROSS, 2003)

Os objetos utilizados são especificados pela *Structure of Management Information* (SMI). A SMI é a linguagem padronizada pela RFC 2578, na versão 2 do SNMP, para definir informações de gerenciamento, e ela basicamente dita os tipos de dados que os objetos utilizam na MIB. (KUROSE; ROSS, 2003)

Por definição as informações na MIB estão agrupadas em módulos, e estes módulos estão estruturados em formato de árvore. Está estrutura foi criada pela *International Organization for Standardization* (ISO). Apesar de toda esta estrutura, apenas as pontas desta árvore, ou seja, os nós da ponta é que são os objetos gerenciados. Cada objeto representa uma informação obtida do dispositivo, como por exemplo, informações sobre utilização da rede e quantidade de portas de rede (KUROSE; ROSS, 2003).

Existem na árvore basicamente três tipos de MIB, a padrão chamada MIB-2 que é descrita pela RFC 1213 que foi criada somente para a gerência de dispositivos de rede ou que envolvam a rede, e ela é fixa e não pode ser modificada. Existem também versões experimentais de MIB construídas para uma aplicação particular, estas versões também são fixas. E por fim existem versões privadas de MIB, e elas têm seu lugar especifico na estrutura da árvore (STALLINGS, 1998). Toda a empresa que desejar criar o seu próprio modelo de informações e disponibilizar em seu dispositivo ou software deve realizar um cadastro junto ao IEEE para obter um numero que a identificará na estrutura da árvore de objetos.

A MIB-2 é a definição da segunda versão da MIB, já que a primeira foi descrita pela RFC 1156. Ela foi criada para superar em vários critérios a sua antecessora, como somente ter objetos com evidências de uso, a possibilidade de ter mais de 100 objetos gerenciáveis, impossibilidade de objetos derivados de outros objetos, objetos de aplicações específicas foram excluídos, entre outros objetivos.

Como a MIB-2 contem apenas objetos considerados essenciais, todos eles são obrigatórios. (STALLINGS, 1998)

A MIB utiliza em sua estrutura interna *Object Identifiers* (OID). Todos os objetos gerenciados possuem um OID único. Eles devem estar aninhados conforme a estrutura padrão da MIB, ou seja, todos os OIDs devem estar em alinhamento numérico compatível com a estrutura da MIB. Abaixo, na figura 1, a estrutura de nomeação de objetos da norma ISO8824 e na figura 2 o desencadeamento da MIB-2.

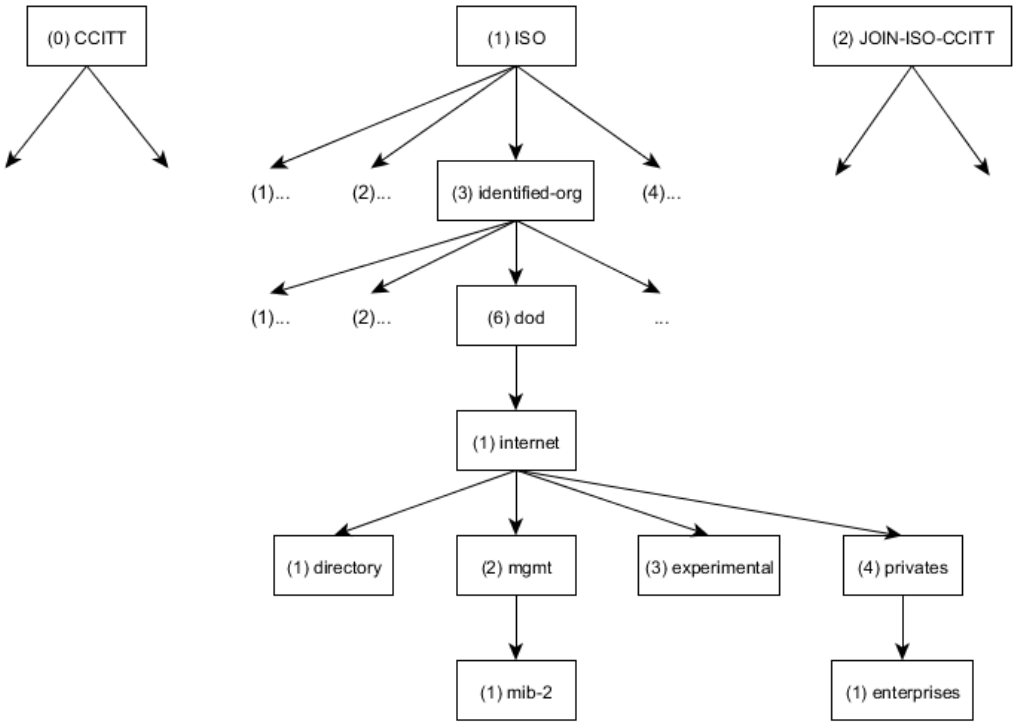


Figura 1 - Estrutura de nomeação de objetos da norma ISO8824.

Fonte: Adaptado de Stallings (2005)

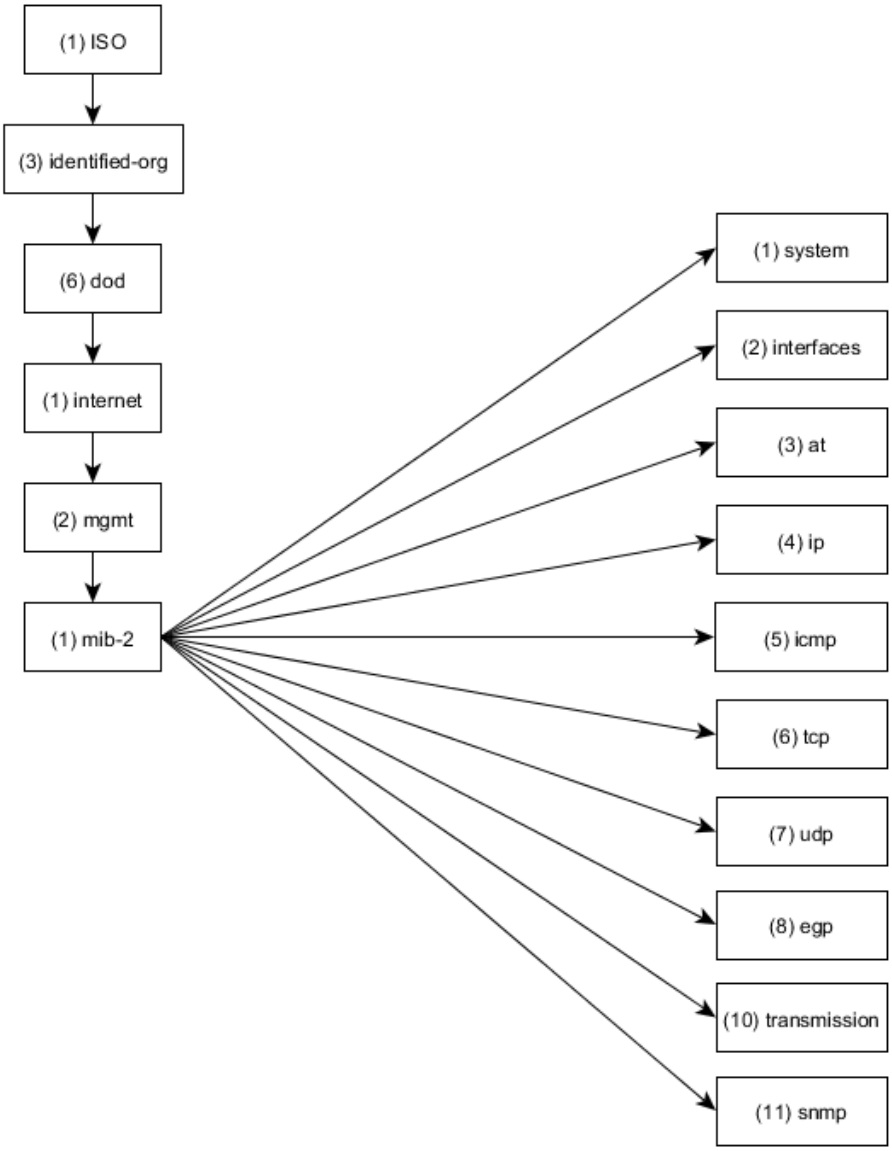


Figura 2 - Estrutura da MIB-2.

Fonte: Adaptado de Stallings (2005)

Com a possibilidade de utilizar objetos específicos na MIB-2, surgiram várias ramificações da sua estrutura. Dentre estas estão a RFC 2863 específica para gerenciar objetos de interfaces de rede, a RFC 1697 para a gerência de objetos de bancos de dados relacionais, a RFC 2790 contendo objetos para gerência dos *hosts* de rede e as RFC 2819 e 4502 que descrevem o *Remote Network Monitoring* (RMON) 1 e 2, este possibilitou a utilização de agentes obtendo dados da rede nas camadas 1 e 2 do modelo *Open Systems Interconnection* (OSI). Outras

ramificações, como para gerência de redes *Frame Relay* e redes *Synchronous Optical Networking* (SONET) também foram criados e ajudaram a perpetuar a MIB-2.

Abaixo, na figura 3, um exemplo da ramificação RFC 2863 para interfaces de rede.

Nome do Objeto	Identificador do Objeto
ifMIB	1.3.6.1.2.1.31
ifMIBObjects	1.3.6.1.2.1.31.1
ifXTable	1.3.6.1.2.1.31.1.1
ifXEntry	1.3.6.1.2.1.31.1.1.1
ifName	1.3.6.1.2.1.31.1.1.1.1
ifHCOutOctets	1.3.6.1.2.1.31.1.1.1.10
ifHCOutUcastPkts	1.3.6.1.2.1.31.1.1.1.11
ifHCOutMulticastPkts	1.3.6.1.2.1.31.1.1.1.12
ifHCOutBroadcastPkts	1.3.6.1.2.1.31.1.1.1.13
ifLinkUpDownTrapEnable	1.3.6.1.2.1.31.1.1.1.14
ifHighSpeed	1.3.6.1.2.1.31.1.1.1.15
ifPromiscuousMode	1.3.6.1.2.1.31.1.1.1.16
ifConnectorPresent	1.3.6.1.2.1.31.1.1.1.17
ifAlias	1.3.6.1.2.1.31.1.1.1.18
ifCounterDiscontinuityTime	1.3.6.1.2.1.31.1.1.1.19
ifInMulticastPkts	1.3.6.1.2.1.31.1.1.1.2
ifInBroadcastPkts	1.3.6.1.2.1.31.1.1.1.3
ifOutMulticastPkts	1.3.6.1.2.1.31.1.1.1.4
ifOutBroadcastPkts	1.3.6.1.2.1.31.1.1.1.5
ifHCInOctets	1.3.6.1.2.1.31.1.1.1.6
ifHCInUcastPkts	1.3.6.1.2.1.31.1.1.1.7
ifHCInMulticastPkts	1.3.6.1.2.1.31.1.1.1.8
ifHCInBroadcastPkts	1.3.6.1.2.1.31.1.1.1.9
ifStackTable	1.3.6.1.2.1.31.1.2
ifStackEntry	1.3.6.1.2.1.31.1.2.1
ifStackHigherLayer	1.3.6.1.2.1.31.1.2.1.1
ifStackLowerLayer	1.3.6.1.2.1.31.1.2.1.2
ifStackStatus	1.3.6.1.2.1.31.1.2.1.3

Figura 3 – Ramificação MIB 2 Interfaces

Fonte: OIDVIEW (2012)

3.1 TIPOS DE DADOS DA MIB

Todos os objetos contidos na MIB são definidos formalmente por um padrão, o padrão utilizado é o *Abstract Syntax Notation One* (ASN.1) definido pelo ITU-T na Recomendação X.680. O ASN.1 define a estrutura dos objetos e a estrutura da MIB em si.

Os objetos que independem do tipo de aplicação estão na classe universal da ASN.1, e que são usados nas MIBs do SNMP estão listados a seguir (ITU-X.680, ano):

Integer: inteiros na faixa de -2^{32} a $2^{31}-1$;

UInteger32: inteiros na faixa de 0 a $2^{32}-1$;

Octet String: *strings* de octetos para dados binários ou textuais arbitrários, pode-se limitar a 255 octetos;

Bit String: enumeração de bits nomeados;

Null: objeto vazio;

Object Identifier: nome administrativamente atribuído a objeto ou outro elemento padronizado, o valor é um sequencia de até 128 inteiros não negativos;

Sequence, Sequence-of: definição de tabelas e suas linhas;

A ASN.1 também define uma classe de aplicação, porem cada aplicação é responsável pelos seus próprios tipos de dados. A RFC 1155 e RFC 2578 listam mais alguns tipos de dados que são utilizados pelas MIBs SNMP, os quais estão listados a seguir (McCLOGHRIE, 1990):

Ipaddress: campo do tipo *octet string* que é representado no formato de endereço *Internet Protocol* (IP) contendo 32 bits;

Counter32: campo de número inteiro (tipo *integer*) não negativo que somente pode ser incrementado até que atinja o seu tamanho máximo, quando então o campo inicia novamente do zero, seu limite é 2^{32};

Counter64: campo de número inteiro (tipo *integer*) não negativo que somente pode ser incrementado até que atinja o seu tamanho máximo, quando então o campo inicia novamente do zero, seu limite é 2^{64};

Gauge32: campo de número inteiro (*integer*) que pode ser incrementado ou decrementado, e quando o campo atinge seu tamanho máximo ele mantem o seu valor até ser ativamente reiniciado, seu limite é 2^{31}-1;

Timeticks: campo de número inteiro (*integer*) que conta o tempo em centésimos de segundos desde uma época definida, seu limite é 2^{32};

Opaque: este tipo de campo tem a capacidade de carregar qualquer tipo de dado, e os dados deste campo são tratados como *octect strings*;

3.2 CRIANDO UMA MIB

Após o entendimento da estrutura e dos tipos de dados utilizados pela MIB, a construção de uma MIB proprietária é relativamente fácil, como mostra Rech Filho (1996).

Basicamente a modelagem de uma MIB é representada em um arquivo de texto puro contendo todos os objetos a serem gerenciados, mas existem vários programas no mercado que dispõem de ferramentas gráficas e intuitivas para a modelagem de MIBs.

Abaixo um trecho do grupo *System* da RFC 1213, onde o *OBJECT-TYPE* é o nome do objeto gerenciado, *SYNTAX* é o tipo de dado contido no objeto, *ACCESS* são as permissões do objeto, *STATUS* se o objeto é obrigatório ou não e se está obsoleto, *DESCRIPTION* é a descrição do objeto e a ultima linha do código é onde está o OID deste objeto dentro do grupo *System*.

```
sysUpTime OBJECT-TYPE
        SYNTAX  TimeTicks
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
```

```
                    "The time (in hundredths of a second) since the
                    network management portion of the system was last
                    re-initialized."
    ::= { system 3 }
```

Tendo em mente o propósito de uma MIB proprietária e escolhendo quais objetos serão criados na mesma, o próximo passo é escrever todos os objetos utilizando o padrão ASN.1. Este passo pode ser considerado como o mais importante e não apenas como uma documentação padronizada, pois o dispositivo gerenciado e o gerente utilizam este arquivo para realizarem seus trabalhos.

Os dispositivos gerenciados, representados por seus agentes, e os gerentes podem utilizar qualquer linguagem de programação, pois ambos compilarão o arquivo de texto da MIB para o formato que eles desejarem. Portanto caso a MIB tenha alguma falha de criação os programas não irão funcionar.

Para facilitar a diagramação da MIB existem vários programas que dispõe de interface gráfica, eles utilizam as conhecidas estruturas em árvore para demonstrar cada objeto gerenciado da MIB. Utilizando aplicativos deste tipo o desenvolvimento da MIB se torna rápido e alguns programas até corrigem erros na lógica da MIB.

Para a utilização no aplicativo gerado ao final desta monografia qualquer MIB pode ser criada, não necessitando o dispositivo existir realmente, pois os testes são realizados apenas no ambiente virtual. Pode-se criar MIBs para geladeiras, bicicletas, caminhões, roteadores, *firewalls*, desde que a lógica e estrutura sejam seguidos.

3.3 OPERAÇÕES DE GERÊNCIA

As operações realizadas do gerente para o agente e vice-versa são descritas genericamente por Stallings (2005) utilizando basicamente três tipos de operações, são elas *Get*, *Set* e *Notify*. Operações do tipo *get* são as operações que solicitam dados alocados no agente para o gerente, operações do tipo *set* servem para definir dados no agente pelo gerente e operações *notify* são utilizadas para que o agente

envie dados não solicitados ao gerente sobre algum evento determinado. (STALLINGS, 2005)

Especificamente o SNMPv2 define sete operações entre gerente e agente. Estas operações são baseadas nos três tipos citados anteriormente (KUROSE; ROSS, 2003). São elas:

GetRequest: o gerente está solicitando ao agente o valor de um OID alocado na MIB do agente;

GetNextRequest; o gerente está solicitando ao agente o valor do próximo OID alocado na MIB do agente;

GetBulkRequest: o gerente está solicitando ao agente todo o bloco de valores posteriores ao OID solicitado, por exemplo, os valores de todos os OID a partir da ramificação 1.3.6.1.2.1.4; na *Protocol Data Unit* (PDU) há uma limitação da quantidade de objetos a serem retornados.

InformRequest: o agente envia ao gerente os OID disponíveis na MIB para que o gerente possa solicitá-los;

SetRequest: o gerente solicita que o agente defina um valor a um OID alocado em sua MIB;

Response: todas as operações acima são respondidas pelo agente com esta resposta indicando inclusive o sucesso ou não da operação solicitada;

SNMPv2-*trap*: o agente ativamente envia um valor ao gerente baseado em um evento excepcional previamente cadastrado em sua MIB.

4 CRIAÇÃO DO AGENTE SNMP

Ao final da criação da MIB num programa gerador de MIB será gerado um arquivo com extensão mib. Apesar da extensão do arquivo o conteúdo é texto puro, onde toda a estrutura e lógica da MIB criada no programa estão no arquivo.

Junto a MIB criada é necessária à criação de outro arquivo que será utilizado como base de dados para a simulação, o qual deve conter os OIDs, os tipos de dados, o tempo de troca entre as posições dos valores e os valores de cada ponto de tempo. Assim o agente desenvolvido poderá simular uma MIB exatamente com os dados que o usuário desejar.

A escolha da linguagem de programação para o desenvolvimento do programa deve levar em consideração dois pontos principais: ser capaz de abrir e manter conexões de rede, estar preparado para trabalhar com o protocolo SNMP e possibilitar a sobreposição de métodos padrões.

Baseado na consideração acima a linguagem escolhida para este trabalho foi o Java, pois o mesmo possui uma extensão para programação de agentes diversos. O Java suporta a sobreposição de métodos, tem licença livre e possibilita um desenvolvimento relativamente rápido de agentes SNMP. (DEITEL, 2003)

4.1 APLICATIVOS SIMILARES

Existem vários programas para a simulação de agentes SNMP, gratuitos e pagos. Os programas pagos normalmente são desenhados para grandes empresas e são de código fonte fechado, o que impossibilita algum tipo de teste mais específico. Já os programas gratuitos são desenhados para possibilitar todo tipo de teste já que a maioria é de código fonte aberto.

Como referência para este trabalho são citados dois programas.

iReasoning SNMP Agent Simulator: é um aplicativo baseado em Java que pode simular agentes SNMPv1/v2c/v3, mostrado na Figura 4. Escrito puramente em

Java, ele pode ser executado em praticamente todas as plataformas. (IREASONING, 2012a)

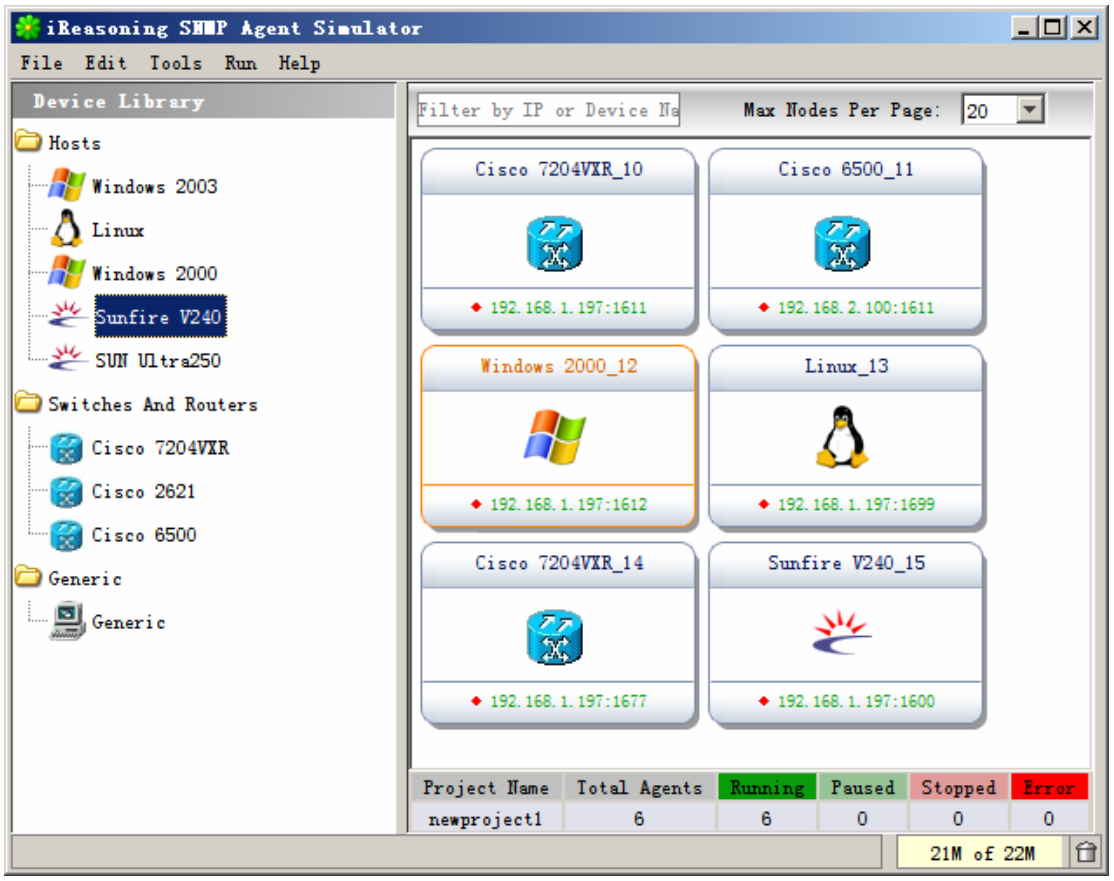


Figura 4 – Tela iReasoning

Fonte: IREASONING (2012b)

SimpleSoft SimpleAgentPro: é um simulador de agentes SNMP com uma interface gráfica fácil de usar, vide a Figura 5, o qual possui capacidade de simular redes complexas.

Usado por mais de 250 empresas, incluindo HP, CA, BMC, IBM e Cisco,

Além de suportar o SNMPv1, v2c e v3, os dispositivos simulados trabalhar com os seguintes protocolos: SSH, HTTP / HTTPS, TL1, TFTP, FTP, DHCP, Netflow.

Os dispositivos podem utilizar MIBs padrão, privadas ou experimentais. O programa possui diagramação de *scripts* que permitem aos dispositivos simulados a aprender e tomar decisões o que pode tornar a simulação de um ambiente mais completa. (SIMPLESOFT, 2012)

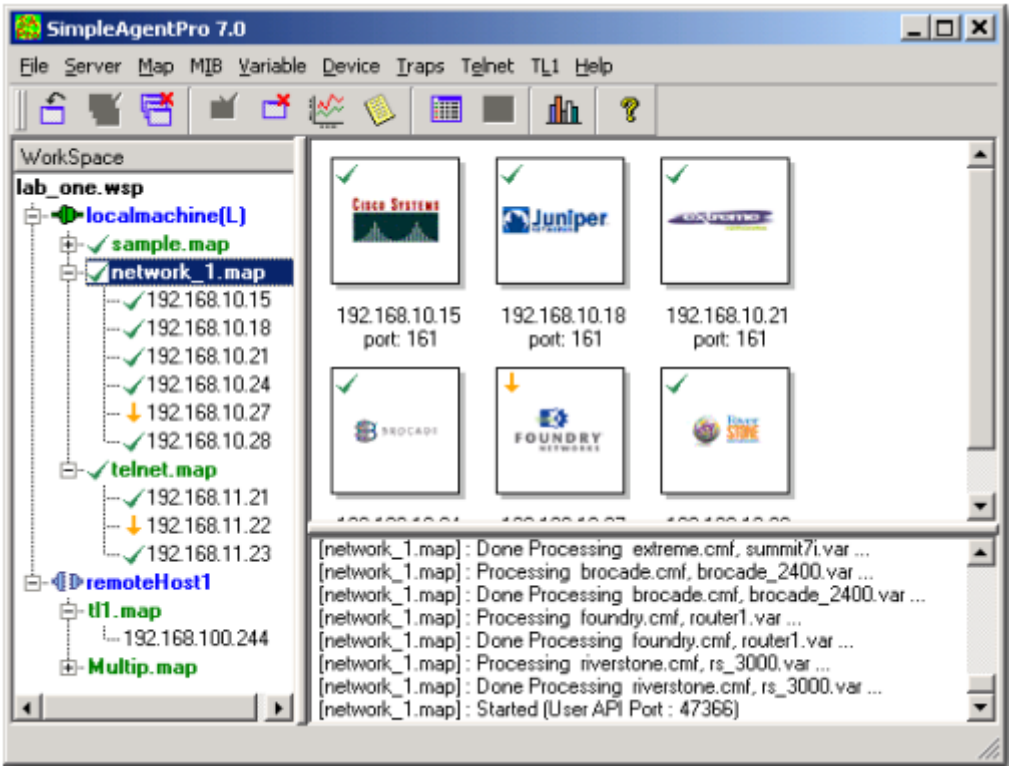


Figura 5 – Tela SimpleSoft

Fonte: SIMPLESOFT (2012)

4.2 APLICATIVO CRIADO

O agente criado é baseado no OpenDMK, um projeto de código aberto baseado no *Java Dynamic Management Kit* (JDMK), que é um kit de ferramentas com tecnologia Java. Este permite aos desenvolvedores criar rapidamente agentes inteligentes baseado no *Java Management Extensions* (JMX). (OPENDMK, 2011)

O Java DMK define uma arquitetura de gerenciamento em três níveis: o nível de instrumentação torna os recursos gerenciáveis como objetos Java, o nível de agente expõe esses objetos para a gerência e o nível de serviços distribuídos permite o acesso remoto e segurança das aplicações.

Ele fornece um modelo distribuído que é independente de protocolo. As aplicações utilizam a API e não se preocupam com o protocolo.

A arquitetura flexível pode distribuir a carga de gerenciamento entre gerentes que podem ser atualizados em tempo real.

Outra possibilidade cogitada para a criação do aplicativo foi o SNMP4J um programa gratuito e de código fonte aberto que implementa o SNMP para Java. Ele suporta a criação de gerentes, bem como de agentes SNMP. Sua API disponibiliza as versões 1, 2c e 3 do SNMP para utilização.

O SNMP4J oferece os seguintes recursos: SNMPv3 com autenticação MD5 e SHA e DES, 3DES, AES 128, AES 192, AES 256 e privacidade. Solicitações síncronas e assíncronas. Código fonte aberto com o modelo de licença Apache. Suporte a multitarefas entre outros. (SNMP4J, 2011)

A escolha pelo OpenDMK se deu devido ao fato dele possuir uma ferramenta própria para compilação de arquivos MIB o mibgen, o qual cria automaticamente todas as classes Java necessárias para utilização no aplicativo. Esta funcionalidade possibilita que o desenvolvimento do aplicativo seja focado apenas nos métodos que precisam ser sobrepostos para que atendam às necessidades do aplicativo criado.

O Apêndice 1 detalha o procedimento de instalação do aplicativo criado e do formato em que o arquivo de simulação deve estar.

4.2.1 PROJETO

Para desenvolver um agente SNMP funcional utilizando o OpenDMK que receba uma MIB de entrada é necessária a criação de um aplicativo anterior ao agente. Este aplicativo será identificado como principal, pois é ele quem receberá as entradas do usuário e criará como produto final outro aplicativo, o agente SNMP.

O aplicativo principal está distribuído em três etapas: a preparação dos fontes, a compilação dos mesmos e a execução do aplicativo Java gerado que é o agente SNMP.

O aplicativo principal recebe três entradas do usuário, mostradas na figura 6, que são: o arquivo contendo a MIB a ser simulada, um arquivo contendo os dados de simulação e a porta em que o agente SNMP deverá rodar.

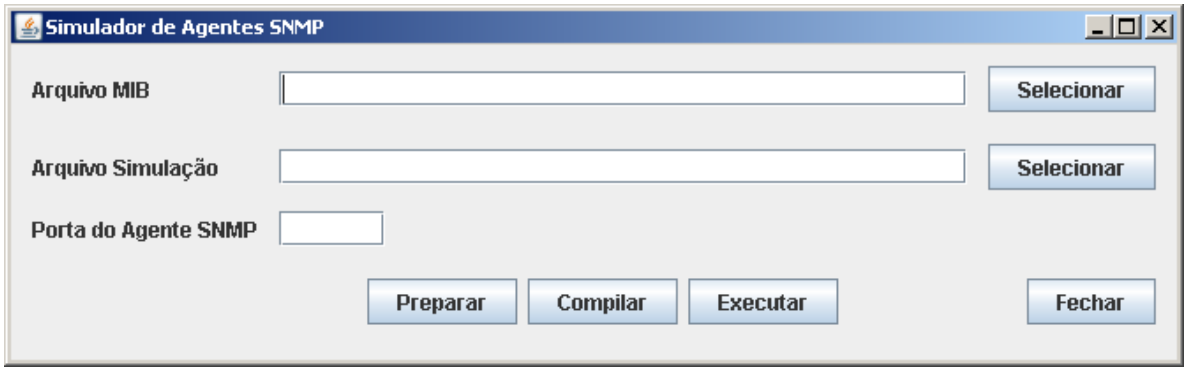


Figura 6 – Tela Primária Simulador

O produto final é o agente que contém a funcionalidade SNMP, ele é um aplicativo Java que é gerado pelo aplicativo principal.

O agente criado visa atender as solicitações mais simples do protocolo SNMP, que são as funções de *get* e *set* de dados na MIB. O método *trap* não está contemplado neste projeto.

Para diminuir a complexidade do desenvolvimento, o agente não utiliza nenhum tipo de banco de dados. Apenas um arquivo de texto contendo os dados a serem simulados é utilizado. Este arquivo é analisado e importado diretamente para uma variável do programa. O agente não valida os dados a serem simulados, como por exemplo, se o OID na MIB é um numero inteiro, mas no arquivo de simulação contém uma *string* o agente simplesmente não retornará nenhuma informação.

4.2.2 DESENVOLVIMENTO

O desenvolvimento também está separado em três etapas, a primeira é a preparação de vários arquivos como o arquivo MIB, o arquivo de simulação, a criação das classes Java pelo gerador de classes mibgen e por final a alteração de alguns métodos nas classes criadas. A segunda é a compilação das classes Java criadas na etapa anterior. E a terceira é execução do agente que foi compilado no passo anterior.

A etapa de preparação, Figura 7, é onde o desenvolvimento se concentra, pois é nela onde são criados todos os arquivos contendo a inteligência de simulação que

são usados nos passos posteriores, nesta etapa são executados os seguintes passos:

- A Inicialização dos diretórios para que todos os diretórios utilizados internamente pela aplicação estejam limpos a cada preparação.
- O arquivo MIB e de simulação inseridos no aplicativo principal são copiados para um diretório interno do aplicativo principal para que sejam alterados e preparados.
- O arquivo da MIB copiado no passo anterior será alterado, o nome original da MIB em *DEFINITIONS* é substituído por um nome conhecido do aplicativo.
- O arquivo de comandos que irá compilar a MIB é criado baseando-se no caminho atual onde o aplicativo principal esta rodando.
- A MIB é compilada ao executar o arquivo de comandos mibgen.bat que foi criado no passo anterior, neste passo são criadas as classes Java necessárias para o agente baseando-se nos itens da MIB.
- As classes Java de objetos escalares devem receber sobreposições especificas de métodos para que utilizem os dados de simulação.
- As classes Java que contem tabelas também devem receber sobreposições especificas de métodos para que utilizem os dados de simulação.
- A classe que inicializa as outras classes criadas é alterada para complementar e substituir métodos que foram criados nos dois passos anteriores.
- A classe principal do agente é copiada e recebe sua única alteração para que utilize a porta de rede inserida no aplicativo principal a ser utilizada pelo agente.
- A lista de classes que é utilizada na compilação é criada contendo todos os arquivos Java que foram criados e modificados nos passos anteriores.

- O arquivo de simulação é mais uma vez copiado, desta vez ele é copiado para o diretório interno do agente e seu nome é alterado para um nome conhecido do agente.

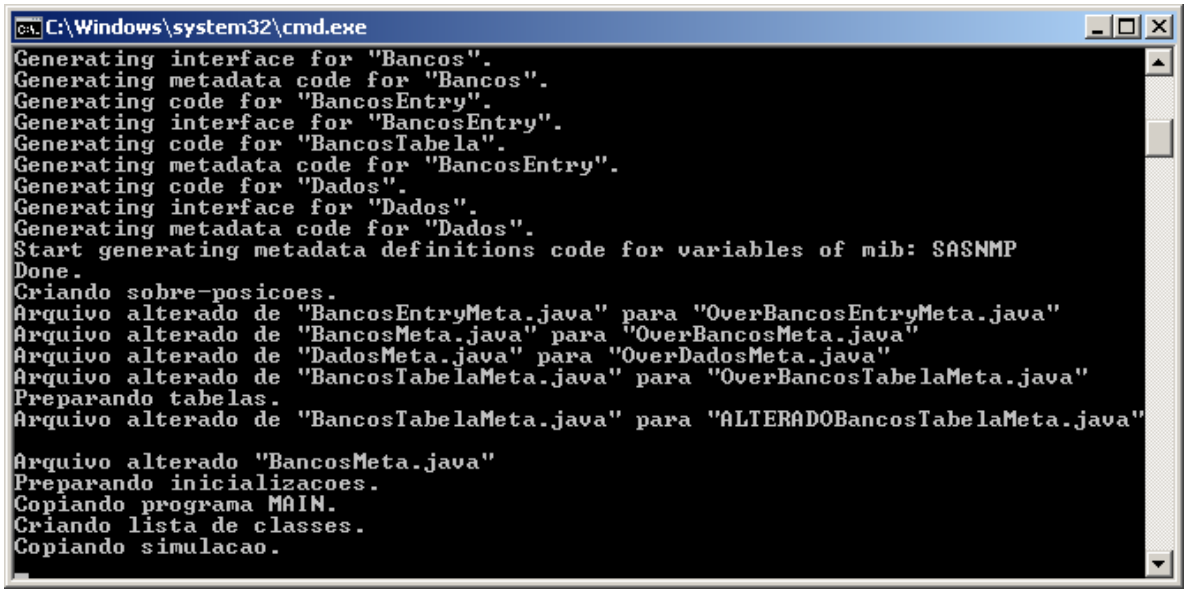


Figura 7 – Tela Preparação Simulador.

A etapa de compilação, Figura 8, irá executar o programa Javac que é o compilador Java. A linha de comando para a compilação é fixa e está dentro de um arquivo de comandos, ela recebe como paramentos dois arquivos, o primeiro é um arquivo fixo que contem as opções para a compilação do aplicativo e o segundo contem a lista de todos os arquivos Java gerados pelo passo de preparação.

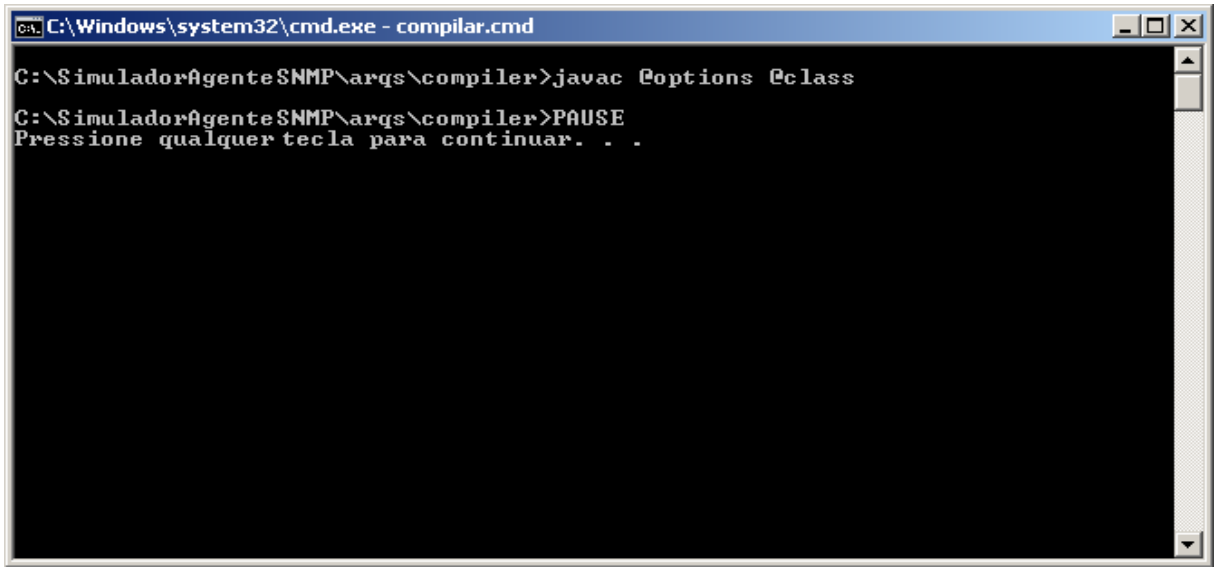


Figura 8 – Tela Compilação Simulador.

A etapa de execução, Figura 9 irá efetivamente iniciar o agente que foi compilado pelo passo anterior. A linha de comando para a execução é fixa e está dentro de um arquivo de comandos, ela recebe como parâmetros o local das bibliotecas JDMK e a classe principal do agente.

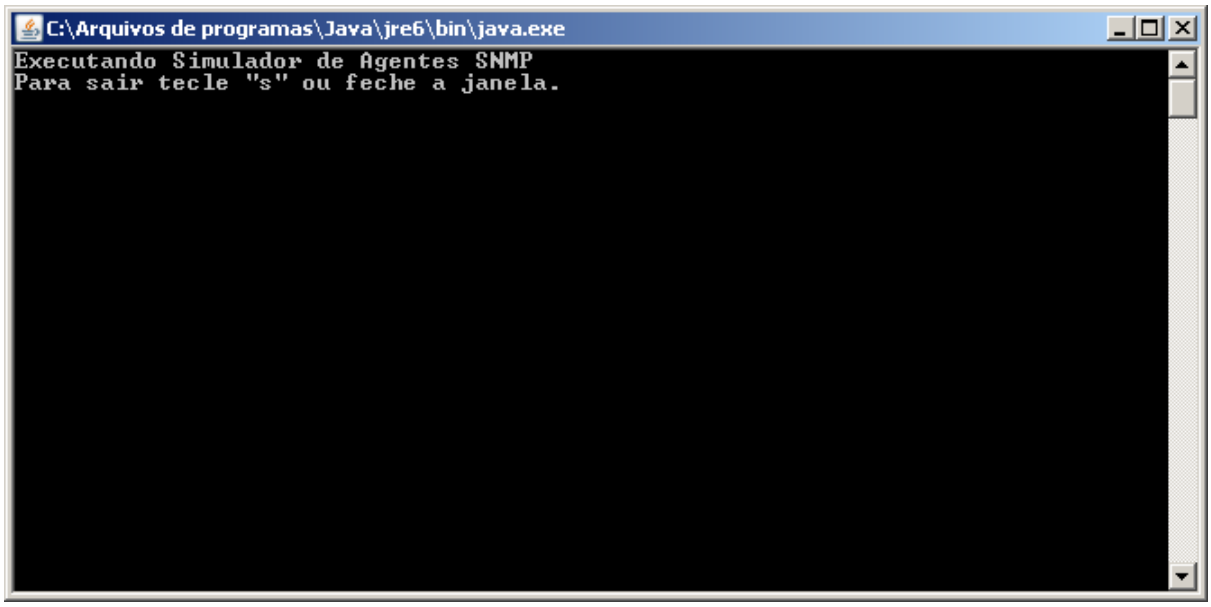


Figura 9 – Tela Execução Simulador.

O método criado para realizar a simulação em unidades de tempo é baseado em um calculo matemático. A simulação não é controlada em tempo como se fosse um relógio, mas sim, a cada solicitação de *get* ou *set* um calculo é feito para determinar em qual posição do vetor de simulação deve ser realizada a operação necessária.

As bases do cálculo são o horário em que o agente foi iniciado, o horário da solicitação e o tempo entre cada simulação, sendo que calculando o primeiro menos o segundo tem-se o tempo passado até a solicitação e então divide-se o resultado pelo terceiro e obtém-se um número inteiro que é índice do vetor de simulação onde a solicitação deve ser executada.

Caso as simulações tenham acabado ou não existam simulações para determinado OID o agente irá retornar para o cliente um valor nulo e exibirá uma mensagem na janela de execução do agente.

Para finalizar o agente basta fechar a janela em que ele esta sendo executado ou basta digitar a letra “s” na janela e teclar *enter* que o agente fechará automaticamente.

4.3 TESTES DO SIMULADOR DE AGENTES

Uma vez o simulador de agentes estando em execução, os dados da MIB objeto de teste podem ser acessados por qualquer programa gerente SNMP, mesmo um simples MIB Browser. Para os testes o simulador foi iniciado na porta 8090 no computador local. Foram realizados testes de *get* e *set* e de alguns erros possíveis.

Exemplo de *GET* nos dados da MIB.

Name/OID	Value	Type
marca.0	Nasa	OctetString
modelo.0	Apolo 17	OctetString
anoFabricacao.0	1973	Integer
anoModelo.0	1975	Integer
eixos.0	4	Integer

Figura 10 – Resultados do MIB Browser.

Fonte: iReasoning (2012)

Exemplo de erro no *GET* quando acabam os dados do arquivo de simulação.

Name/OID	Value	Type
temCobrador.0		Null

Figura 11 – Resultado *GET* sem dados do MIB Browser.

Fonte: iReasoning (2012)

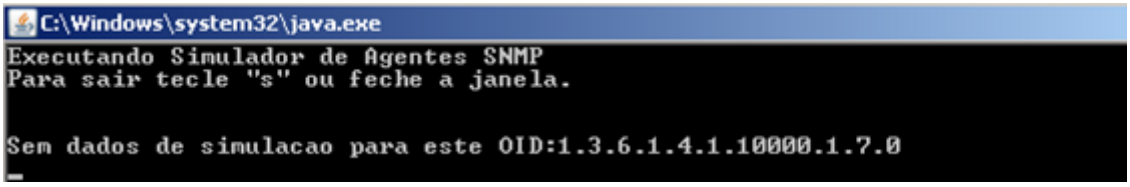


Figura 12 – Resultado *GET* sem dados do Simulador.

Exemplo de erro no *GET* em dados incorretos no arquivo de simulação, por exemplo: o campo é numérico mas no arquivo contém texto.

Figura 13 – Resultado *GET* com erro do MIB Browser.

Fonte: iReasoning (2012)

Exemplo de *SET.*

Figura 14 – Resultado *SET* do MIB Browser.

Fonte: iReasoning (2012)

Exemplo de erro no *SET* em dado do arquivo de simulação ocorre quando a simulação para o dado acabou.

Figura 15 – Resultado *SET* com erro do MIB Browser.

Fonte: iReasoning (2012)

5 CONCLUSÃO

Atualmente o protocolo SNMP é líder neste segmento de mercado, ele esta disponível em praticamente todo dispositivo gerenciável de rede, isto se deve ao fato de apesar de ser um protocolo antigo ele não parou no tempo e suas novas versões têm carregado consigo muitos anos de experiência.

Muitas empresas antes de implantar suas redes utilizam aplicativos de simulação de redes, e estes aplicativos possibilitam a simulação de dispositivos SNMP e até possibilitam que o usuário utilize seus próprios testes de simulação. Estes aplicativos atestam a grande utilidade do protocolo no gerenciamento de redes.

Normalmente estes aplicativos são produtos comerciais com código fonte fechado e não permitem que seja utilizada uma MIB criada pela empresa. Para poder simular seu próprio dispositivo a empresa tem de desenvolver seu próprio simulador, o que para alguns protocolos pode ser lento e problemático, mas, para o SNMP é rápido e fácil.

O SNMP tem vários pacotes prontos para utilização em várias linguagens de programação. Existem alguns aplicativos livre e de código fonte livre em que as empresas podem suprir suas necessidades. Por serem livres caso a empresa tenha alguma necessidade específica ela mesma pode desenvolver sua solução.

No meio acadêmico o simulador ajuda no estudo da gerência de redes e do protocolo SNMP, pois o aprendizado teórico pode ser colocado em prática e isto aumenta a percepção do aluno sobre o funcionamento do protocolo e da rede em si.

O aplicativo desenvolvido neste trabalho busca tirar o foco de que equipamentos que utilizam redes são computadores e os equipamentos de rede, pois o aplicativo pode receber qualquer MIB que o usuário criar, como de um ônibus, ou uma bicicleta. Abre-se a possibilidade de gerenciar simuladamente qualquer objeto que se queira, abrindo o pensamento para futuras aplicações da gerência pelo protocolo SNMP.

A utilização da linguagem de programação Java que hoje é ensinada na maioria dos cursos superiores de informática facilitou o desenvolvimento do aplicativo, pois com sua extensão JDMK pronta para utilização com o protocolo SNMP e o compilador de MIB mibgen o trabalho teve foco no desenvolvimento das necessidades do agente sugerido e não no desenvolvimento do protocolo SNMP ou de suas funcionalidades em si.

Por fim, esse esforço serve para mostrar como o gerenciamento via protocolo SNMP pode ser facilmente simulado, e no agente sugerido pelo trabalho pode-se utilizar qualquer dispositivo que imaginariamente possa ser gerenciado. E o futuro das redes de computadores e sua conexão com diversos dispositivos diferentes mostra uma grande janela de possibilidades no estudo da gerência de redes.

Em trabalhos futuros pode-se implementar a função *trap*, aprimorar o sistema de mensagens ao usuário para que ele possa melhor acompanhar o que o agente está executando e utilizar a versão 3 do protocolo SNMP utilizando todas as funcionalidades de segurança disponíveis.

REFERÊNCIAS

DEITEL, H. M. **JAVA Como Programar**. 4. ed. [S.l.]: Editora Bookman, 2003.

DMTF, **Web-Based Enterprise Management FAQs**. Disponível em < http://www.dmtf.org/about/faq/wbem_faq>. Acesso em 10 de janeiro de 2012.

IREASONING, **Simulator Screenshot**. Disponível em <http://ireasoning.com/snmpsimulator.shtml>. Acesso em 12 de janeiro de 2012a.

IREASONING, **Simulator Screenshot**. Disponível em <http://ireasoning.com/simulatorscreenshot.shtml>. Acesso em 12 de janeiro de 2012b.

KUROSE, J. F.; ROSS, K. W. Redes de Computadores e a Internet: uma nova

abordagem. Tradução de Arlete Simille Marques. São Paulo: Addison Wesley, 2003. cap.6, p. 377-440.

McCLOGHRIE, K. Structure and identification of management information for TCP/IP-based internets - RFC-1155, IETF, 1990.

OIDVIEW, **MIB IF-MIB**. Disponível em <http://www.oidview.com/mibs/0/IF-MIB.html>. Acesso em 12 de janeiro de 2012.

OPENDMK, **Project OpenDMK**. Disponível em <http://opendmk.Java.net/>. Acesso em 10 de outubro de 2011.

RECH FILHO, A. **Estudos para a implantação de uma gerência de rede corporativa utilizando arquitetura de protocolos abertos**. 1996. 147 f. Dissertação (Mestrado em Engenharia Elétrica e Informática Industrial) - Centro Federal de Educação Tecnológica do Paraná, Curitiba-PR, 1996.

SAYDAM, T; MAGEDANZ, T. From Networks and Network Management into Service and Service Management, **Journal of Networks and System Management**, volume 4, 1996.

SIMPLESOFT, **Simple Agent Pro**. Disponível em http://www.snmptest.com/datasheets/simpleagentpro.pdf. Fig 1. Acesso em 12 de janeiro de 2012.

SNMP4J, **The SNMP API for Java**. Disponível em <http://www.snmp4j.org/>. Acesso em 10 de outubro de 2011.

STALLINGS, W. **SNMP, SNMPv2, SNMPv3, and RMON 1 and 2**. 3rd Ed. Reading: Addison-Wesley, 1998.

STALLINGS, W. Redes e Sistemas de Comunicação de Dados: teoria e

aplicações corporativas. 5. ed. Rio de Janeiro: Campus, 2005.

APÊNDICE 1

MANUAL SIMULADOR DE AGENTE SNMP

REQUISITOS

Sistema Operacional Windows XP (ou maior).

Java JDK 1.6 (ou maior).

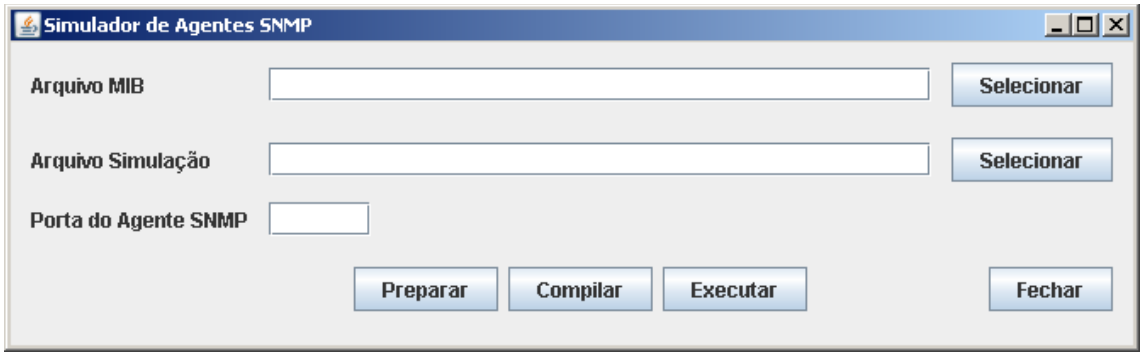
Variável de ambiente PATH deve conter o diretório bin da instalação do Java JDK.

INSTALAÇÃO

Copiar o arquivo SimuladorAgenteSNMP.zip e descompactá-lo.

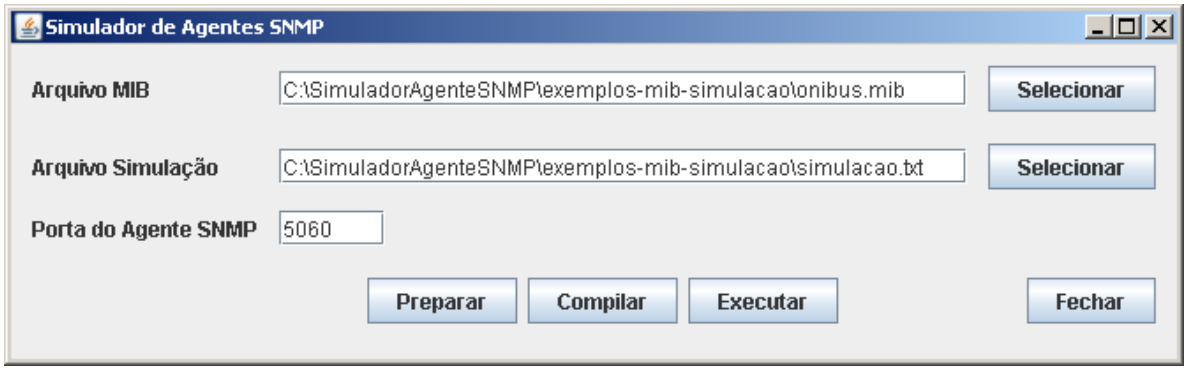
INICIANDO O APLICATIVO

Para rodar o aplicativo basta dar 2 cliques no arquivo “SimuladorAgenteSNMP.bat”. As seguintes janelas se abrirão.



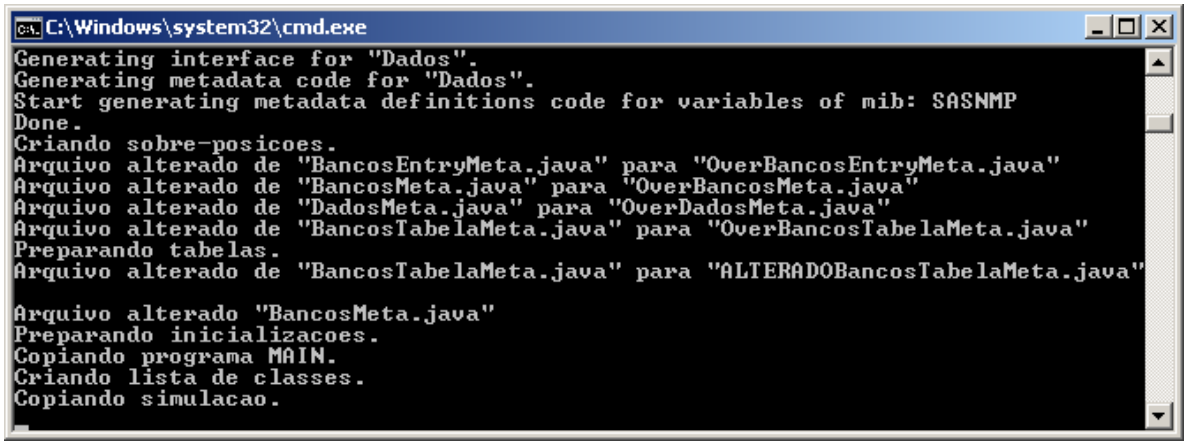
UTILIZANDO O APLICATIVO

Selecione o arquivo “.mib”, um arquivo de texto contendo a simulação e insira uma porta para o agente, como no exemplo abaixo.

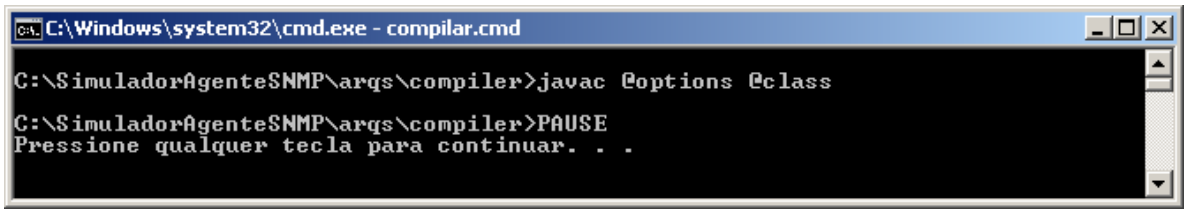


Os passos **Preparar**, **Compilar** e **Executar** devem sempre ser executados na ordem proposta a seguir.

1) Clicando em **Preparar**, o aplicativo irá realizar as operações necessárias para a criação do código fonte do agente, a janela inicial será preenchida com dados da preparação.



2) Clicando em **Compilar**, o aplicativo executará a compilação dos códigos fonte criados no passo anterior. Uma janela como a abaixo se abrirá. Pressione qualquer tecla para continuar, a janela se fechará.



3) Para iniciar o agente SNMP compilado clique em **Executar**. Uma janela como a abaixo se abrirá indicando que o agente está operacional.

FORMANTO DO ARQUIVO DE SIMULAÇÃO

O arquivo de simulação deve ser texto puro e estar no formato a seguir:

OID | Tempo de troca entre os valores | Valores separados por pipe

1.3.6.1.4.1.10000.1.2.0|30|Apolo 17| Apolo 13| Apolo 11| Apolo 7|Apolo

Caso queira que o valor seja constante coloque o tempo de sequencia como 0 (zero).

1.3.6.1.4.1.10000.1.1.0|0|Nasa