FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Dokumentace k projektu

Rozšíření SNMP agenta

Síťové aplikace a správa sítí

15. listopadu 2020 Mirka Kolaříková(xkolar76)

Obsah

1 Zadání projektu

Cílem projektu je implementace vlastního MIB modulu a dynamicky načítatelné rozšíření SNMP agenta net-snmp v jazyce C.

2 Úvod do problematiky

Simple Network Management Protocol (SNMP) je jednoduchý a snadno implementovatelný protokol využívaný pro správu sítí. Slouží k získávání a nastavování informací mezi řídící jednotkou a agentem na monitorovaném zařízením. Podporu SNMP má velké množství aktivních síťových prvků.

Řídící jednotka je serverová aplikace, která posílá požadavky, sbírá a zpracovává data pro monitorování sítě.
Agent na monitorovaném zařízení je aktivní proces běžící na monitorovaném zařízení, který sbírá provozní informace a odpovídá na dotazy řídící jednotce a generuje trap zprávy.

SNMP obsahuje tři základní typy příkazů:

- **get**: řídící jednotka si žádá hodnoty daného objektu od agenta
- **set**: řídící stanice nastaví hodnotu daného objektu na agentovi
- **trap**: agent asynchronně oznamuje řídící jednotce, že nastala nějaká neočekávaná situace

MIB databáze monitorovaných objektů je soubor všech objektů na monitorovaném zařízení. Každý objekt reprezentuje nějakou sledovanou informaci. Řídící jednotka zašle požadavek agentovi na zápis nebo čtení z objektu, agent požadavek provede a odešle řídící jednotce odpověď. Řídící stanice tak může díky pravidelným dotazům monitorovat stav sítě. [9]

3 Implementace

3.1 definice objektů v MIB

MIB je psána v jazyce SMI (Structure Management Information) podle standartu RFC 1155 [8], který definuje pravidla, jak vytvářet objekty SNMP. Každý objekt má jméno dané jednoznačným identifikátorem OID [3].

MIB pro tento projekt se jmenuje **MY-MIB** a obsahuje modul **isa-project-module** se čtyřmi objekty. MIB je podle zadání registrována pod OID **.1.3.6.1.3** (iso.org.dod.internet.experimental) s vlastním číslem **22**. Obsahuje informace o názvu modulu, poslední update, organizaci, kontaktní informace a popis modulu. Definice tohoto modulu je následující:

```
isa-project-module MODULE-IDENTITY
        LAST-UPDATED "202010260000Z"
        ORGANIZATION "VUT-FIT"
        CONTACT-INFO "email : xkolar76@stud.fit.vutbr.cz"
        DESCRIPTION  "MIB for ISA school project"
        ::= { experimental 22 }
```

isa-project-module obsahuje následující 4 objekty:

- **.1.3.6.1.3.22.1 `loginString`**: read-only string obsahující login xkolar76
- **.1.3.6.1.3.22.2 `timeString`**: read-only string, s aktuálním UTC časem
- **.1.3.6.1.3.22.3 `readwriteInt`**: read-write integer, defaultní hodnota je 666
- **.1.3.6.1.3.22.4 `operatingSystem`**: read-only string obsahující typ operačního systému

Každému objektu je kromě názvu a čísla OID v MY-MIB definováno:

- **SYNTAX** určující datový typ objektu, v MY-MIB je to `DisplayString` nebo `Integer32`
- **MAX-ACCES** určuje, jak k proměnné přistupovat, v MY-MIB `read-only` nebo `read-write`
- **STATUS** určující stav proměnné, v MY-MIB je u všech objektů `current`
- **DESCRIPTION** krátký popis objektu
- **DEFVAL** defaultní hodnota

Samotná definice objektu (tady objektu **.1.3.6.1.3.22.4**) potom vypadá takto:

```
operatingSystem OBJECT-TYPE
    SYNTAX              DisplayString
    MAX-ACCESS          read-only
    STATUS              current
    DESCRIPTION         "OS"
    DEFVAL             { "CentOS Linux 7 (Core)" }
    ::= { isa-project-module 4 }
```

Vytvořený MIB je potom umístěn do složky `/usr/share/snmp/mibs`.

3.2 Vytvoření zdrojových souborů

Vytvořený MIB je načtený do terminálu příkazem: `export MIBS="+MY-MIB"`. Pomocí programu **mib2c** [1] lze vygenerovat šablonu zdrojových souborů na základě modulu definovaného v MIB podle určité konfigurace. Vytvoří .c a .h soubory, kde má každý objekt z modulu připravený handle pro každý objekt. Vybraný konfigurační soubor je mib2c.scalar.conf.
Příkaz byl použit takto:
`mib2c -c mib2c.scalar.conf isa-project-module`.

Vygenerované soubory jsou `isa_project_module.c` a `isa_project_module.h`. Ty je poté potřeba upravit a implementovat potřebné části kódu.

3.3 Objekt timeString .1.3.6.1.3.22.2

Objekt **.1.3.6.1.3.22.2** je read-only string, který vrací UTC čas naformátovaný podle RFC 3339 [6].

Aktuální čas je při každém volání zjištěn pomocí funkce **time()** a poté převeden na UTC čas funkcí **gmtime()**. Takto zjištěný čas je poté naformátován pomocí funkce **strftime()** a uložen do stringu. Všechny tyto tři funkce jsou z knihovny **time.h**.
Část kódu, která zjistí aktuální čas a naformátuje jej podle RFC 3339:

```
time_t current_time = time(&current_time);
struct tm *ptm = gmtime(&current_time); //UTC time
char buffer[21];
strftime(buffer,21,"%FT%TZ", ptm);
```

3.4 Objekt operatingSystem .1.3.6.1.3.22.4

Objekt **.1.3.6.1.3.22.4** je read-only string, který vrací typ operačního systému. Program na základě makra, který překladač definoval při překladu, zjistí typ operačního systému a nahraje jej do stringu [7]. Program rozpozná 4 hlavní operační systémy Windows, Linux, Apple a BSD. V případě jiného operačního systému je do stringu nahráno ”other”.

Část kódu, která zjistí operační systém podle makra:

```
#if _WIN32
        static char os[] = ”Windows”;
#elif __linux__
        static char os[] = ”Linux”;
#elif __APPLE__
        static char os[] = ”Apple”;
#elif BSD
        static char os[] = ”BSD”;
#else
        static char os[] = ”other”;
#endif
```

4 Testování

4.1 Testování MIB

Pro zkontrolování syntaxe byl použit MIB validátor.

MY-MIB byla načtena do terminálu pomocí příkazu `export MIBS=”+MY-MIB”` , a poté otestována pomocí utility **snmptranslate** [2] , zda má správnou strukturu a zda je správný překlad jmen objektů na OID a naopak. Příklady použití snmptranslate:

- `snmptranslate -Tp -IR isa-project-module`
- `snmptranslate .1.3.6.1.3.22.1`
- `snmptranslate -On MY-MIB::loginString.0`

```
[user@localhost Downloads]$ snmptranslate -Tp -IR isa-project-module
+--isa-project-module(22)
   +-- -R-- String    loginString(1)
   |        Textual Convention: DisplayString
   |        Size: 0..255
   +-- -R-- String    timeString(2)
   |        Textual Convention: DisplayString
   |        Size: 0..255
   +-- -RW- Integer32 readwriteInt(3)
   +-- -R-- String    operatingSystem(4)
            Textual Convention: DisplayString
            Size: 0..255
[user@localhost Downloads]$
```

Obrázek 1: Struktura objektů

```
[user@localhost Downloads]$ snmptranslate .1.3.6.1.3.22.1
MY-MIB::loginString
[user@localhost Downloads]$ snmptranslate .1.3.6.1.3.22.2
MY-MIB::timeString
[user@localhost Downloads]$ snmptranslate .1.3.6.1.3.22.3
MY-MIB::readwriteInt
[user@localhost Downloads]$ snmptranslate .1.3.6.1.3.22.4
MY-MIB::operatingSystem
[user@localhost Downloads]$
```

Obrázek 2: Překlad OID na jméno

```
[user@localhost Downloads]$ snmptranslate -On MY-MIB::loginString.0
.1.3.6.1.3.22.1.0
[user@localhost Downloads]$ snmptranslate -On MY-MIB::timeString.0
.1.3.6.1.3.22.2.0
[user@localhost Downloads]$ snmptranslate -On MY-MIB::readwriteInt.0
.1.3.6.1.3.22.3.0
[user@localhost Downloads]$ snmptranslate -On MY-MIB::operatingSystem.0
.1.3.6.1.3.22.4.0
[user@localhost Downloads]$
```

Obrázek 3: Překlad jména objektu na OID

4.2 Testování snmpget a snmpset

Modul isa_project_module byl načten do agenta[4], viz sekce 5. Poté bylo u všech objektů testováno snmpget a u objektu readwriteString i snmpset.

```
[user@localhost Downloads]$ snmpget localhost  MY-MIB::loginString.0
MY-MIB::loginString.0 = STRING: xkolar76
[user@localhost Downloads]$ snmpget localhost  MY-MIB::readwriteInt.0
MY-MIB::readwriteInt.0 = INTEGER: 666
[user@localhost Downloads]$ snmpget localhost  MY-MIB::timeString.0
MY-MIB::timeString.0 = STRING: 2020-11-18T11:42:04Z
[user@localhost Downloads]$ snmpget localhost  MY-MIB::operatingSystem.0
MY-MIB::operatingSystem.0 = STRING: Linux
[user@localhost Downloads]$
```

Obrázek 4: snmpget nam všemy objekty

```
[user@localhost Downloads]$ snmpset localhost  MY-MIB::readwriteInt.0 i 80
MY-MIB::readwriteInt.0 = INTEGER: 80
[user@localhost Downloads]$ snmpget localhost  MY-MIB::readwriteInt.0
MY-MIB::readwriteInt.0 = INTEGER: 80
[user@localhost Downloads]$ snmpset localhost  MY-MIB::readwriteInt.0 i 555
MY-MIB::readwriteInt.0 = INTEGER: 555
[user@localhost Downloads]$ snmpget localhost  MY-MIB::readwriteInt.0
MY-MIB::readwriteInt.0 = INTEGER: 555
[user@localhost Downloads]$
```

Obrázek 5: snmpset nad objektem readwriteInt

5 Návod na použití

1. MY-MIB.txt dáme do složky `/usr/share/snmp/mibs` a načteme ji do terminálu pomocí `export MIBS="+MY-MIB"` a ve složce se zdrojovými soubory spustíme překlad pomocí `make`.
2. V prvním okně spustíme daemon pro SNMPD agenta s debugovacími přepínači:
   `% sudo snmpd -f -L -DnstAgentPluginObject,dlmod`
3. V druhém okně pomocí příkazu snmpset vytvoříme nový řádek v dlmod tabulce:
   `% snmpset localhost UCD-DLMOD-MIB::dlmodStatus.1 i create`
4. Vlastnosti řádku nastavíme tak, aby ukazoval na náš objekt a dáme mu jméno:
   `% snmpset localhost UCD-DLMOD-MIB::dlmodName.1 s "isa_project_module"`
   `UCD-DLMOD-MIB::dlmodPath.1 s "/path/to/isa_project_module.so"`
5. Načteme shared object soubor do běžícího agenta:
   `% snmpset localhost UCD-DLMOD-MIB::dlmodStatus.1 i load`
6. Nyní můžeme nad modulem provádět příkazy snmpget a snmpset [5].

Příklady použití:

- `snmpget localhost MY-MIB::operatingSystem.0`
- `snmpget localhost MY-MIB::timeString.0`
- `snmpget localhost .1.3.6.1.3.22.1.0`
- `snmpset localhost MY-MIB::readwriteInt.0 i 89`
- `snmptranslate -On MY-MIB::loginString.0`
- `snmptranslate .1.3.6.1.3.22.2`

6 Závěr

Tento projekt byl zajímavý a nebyl Implementovala jsem všechny specifikace zadání projektu a při testování jsem nenarazila na žádný problém.

Literatura

[1] mib2c. [online]. Dostupné z: http://www.net-snmp.org/docs/man/mib2c.html

[2] NET-SNMP Tutorial – snmptranslate. [online], 2020. Dostupné z: http://net-snmp.sourceforge.net/tutorial/tutorial-5/commands/snmptranslate.html

[3] Net-SNMP Tutorial – MIB Module. [online], květen 2020. Dostupné z: http://www.net-snmp.org/tutorial/tutorial-5/toolkit/mib_module/

[4] Agent Architecture. [online], listopad 2015. Dostupné z: http://www.net-snmp.org/wiki/index.php/Agent_Architecture

[5] TUT:Writing a Dynamically Loadable Object. [online], Srpen 2010. Dostupné z: http://net-snmp.sourceforge.net/tutorial/tutorial-5/commands/snmptranslate.html

[6] G. Klyne, C. N.: Date and Time on the Internet: Timestamps. [online], červenec 2002. Dostupné z: https://tools.ietf.org/html/rfc3339

[7] IQ, O.: Detect Operating System in C. [online], 2020. Dostupné z: https://iq.opengenus.org/detect-operating-system-in-c/

[8] M. Rose, K. M.: Structure and Identification of Management Information for TCP/IP-based Internets. [online], květen 1990. Dostupné z: https://tools.ietf.org/html/rfc1155

[9] Matoušek, P.: *Síťové aplikace a jejich architektura*, kapitola Architektura SNMP. Brno: VUTIUM, 2014, ISBN 978-80-214-3766-1.