VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Rozšíření SNMP agenta

18. listopadu 2020

Martin Kostelník

Obsah

1 Úvod

Simple Network Management Protocol (dále SNMP) je síťový protokol pro správu a monitorování zářízení připojených do sítě. Úkolem bylo vytvořit a implementovat dynamicky načítatelný modul, který bude sloužit jako rozšíření běžícího SNMP agenta. SNMP agent je aplikace běžící na monitorovaném zařízení, která sbírá informace o jednotlivých prvcích systému. Implementovaný modul by měl takového agenta rozšířit o čtyři objekty obsahující informace různých druhů.

2 Popis implementace

Aplikace je implementována jako dynamicky načítatelný modul. Jako první byl vytvořen textový soubor obsahující definici Management information base (dále MIB). Tento soubor obsahuje celkem pět objektů. První je samotný modul, který se do SNMP stromu řadí pod větev experimental a má vlastní číslo 22. Následují čtyři skalární objekty obsahující užitečná data. Jednotlivé objekty i následný přístup k datům je detailně popsán v kapitole 4.

Pro vygenerování koster zdrojových souborů byl použit nástroj `mib2c` [2]. Tento nástroj z textového souboru obsahující definici MIB vytvořil dva soubory. Hlavičkový soubor `ISAMIBModule.h` a zdrojový soubor `ISAMIBModule.c`. Hlavičkový soubor nebylo nutno již nijak upravovat. Zdrojový soubor však upraven být musel. Byly vytvořeny globální statické proměnné, ve kterých jsou uloženy data našich objektů. Pro read-write objekt bylo nutno vytvořit ještě proměnnou na zálohování hodnoty, pokud by při přepisování došlo k chybě.

Při načtení modulu do běžícího agenta se musí modul nejprve inicializovat. Inicializaci zprostředkovává funkce `init_ISAMIBmodule()`. Tato funkce byla vygenerována kompletní a nemusela být upravena.

Dále existují funkce na manipulaci s jednotlivými objekty. U každé funkce byly upraveny části, která zprostředkovává čtení dat z objektu. Dále bylo implementováno získávání aktuálního času ve funkci manipulující s objektem, který čas specifikuje. UTC čas je získán pomocí knihovní funkce `gmtime()` a následně zformátován pomocí funkce `sprintf()` dle standardu RFC 3339 [6]. Časový formát vypadá následovně: `YYYY-MM-DDTHH:MM:SSZ`. U funkce pro objekt s číslem bylo implementováno navíc nastavení proměnné na hodnotu zadanou uživatelem včetně zálohování pro případ chyby při zápisu.

Pro řízení překladu je použit program make. Makefile obsahuje kromě cíle pro překlad také cíl `clean`, který odstraní soubory vzniklé při překladu a cíl `pack`, který projekt zabalí do formátu `tar`, ve kterém je projekt odevzdán.

3 Spuštění aplikace

Nejprve je nutné aplikaci přeložit příkazem `make` v adresáři s projektem. Tím nám vznikne binární soubor `ISAMIBModule.so`. Dále zkopírujeme soubor s MIB definicí do adresáře `/usr/share/snmp/mibs`. Ke spuštění aplikace budeme potřebovat dvě okna s terminálem. V jednom okně terminálu spustíme SNMP agenta příkazem:

```
snmpd -f -L
```

Pokud spuštění agenta proběhlo v pořádku, měli bychom být uvítáni zprávou o použité verzi agenta. Nyní se přepneme do druhého okna terminálu, kde načteme náš modul do již běžícího agenta. Toho dosáhneme následující sekvencí třech příkazů:

```
snmpset localhost UCD-DLMOD-MIB::dlmodStatus.1 i create
```

`snmpset localhost UCD-DLMOD-MIB::dlmodName.1 s "ISAMIBModule" UCD-DLMOD-MIB::dlmodPath.1 s "path"`, kde `path` je absolutní cesta k binárnímu souboru s modulem

```
snmpset localhost UCD-DLMOD-MIB::dlmodStatus.1 i load
```

První příkaz vytvoří nový záznam v dlmod tabulce. Druhý nastaví název modulu a cestu k němu do vytvořeného řádku tabulky. Třetí příkaz už konečně načte náš modul do běžícího agenta. Jak jeho funkcionalitu vyzkoušet je popsáno v následující kapitole.

4 Objekty modulu

V modulu jsou implementovány čtyři různé objekty.

4.1 loginObject

Tento objekt je se registruje pod OID .1.3.6.1.3.22.1. Je implementován jako read-only textový řetězec, konkrétně datovým typem DisplayString. Objekt obsahuje autorův login do informačního systému FIT VUT. Hodnotu můžeme zjistit příkazem `snmpget` takto:

```
snmpget localhost ISA-MIB::loginObject.0
```

Na výstupu terminálu bychom měli vidět řádek:

```
ISA-MIB::loginObject.0 = STRING: xkoste12
```

4.2 timeObject

Objekt je registrován pod OID .1.3.6.1.3.22.2. Jako předchozí objekt je i tento implementován jako read-only textový řetězec, typ DisplayString. Jeho obsahem je aktuální UTC čas formátovaný dle standardu RFC 3339 [6].

```
snmpget localhost ISA-MIB::timeObject.0
```

Na výstupu terminálu uvidíme řádek s aktuálním UTC časem. V čase testování:

```
ISA-MIB::timeObject.0 = STRING: 2020-11-18T17:45:59Z
```

4.3 intObject

U toho objektu je kromě čtení povolen i zápis. Registruje se pod OID .1.3.6.1.3.22.3. Na rozdíl od předešlých objektů je implementován jako celočíselný typ (Integer32). V následujících příkladech je ukázáno, jak hodnotu z objektu číst, ale i jak hodnotu zapsat.

```
snmpget localhost ISA-MIB::intObject.0
```

Na výstupu terminálu můžeme vidět výchozí hodnotu objektu, která je 0:

```
ISA-MIB::intObject.0 = INTEGER: 0
```

Změníme hodnotu z 0 na 42 příkazem `snmpset` následovně:

```
snmpset ISA-MIB::intObject.0 i 42
```

Když nyní znovu získáme hodnotu objektu pomocí příkazu `snmpget` stejným způsobem jako dříve, uvidíme novou hodnotu:

```
ISA-MIB::intObject.0 = INTEGER: 42
```

4.4 ramObject

Objekt v sobě uchovává množství operační paměti systému v MB. OID tohoto objektu je .1.3.6.1.3.22.4. Pro získání informace o množství paměti je použita funkce `sysinfo()` z hlavičkového souboru `sysinfo.h`. Z tohoto důvodu je program plně funkční pouze na operačním systému Linux. Při použití na jiných operačních systémech bude hodnota objektu vždy nula.

```
snmpget localhost ISA-MIB::ramObject.0
```

Na výstupu terminálu se zobrazí množství operační paměti systému v MB. Pro virtuální stroj použitý při realizaci projektu je výstup:

```
ISA-MIB::ramObject.0 = INTEGER: 1837
```

Literatura

[1] TUT:Writing a Dynamically Loadable Object. [online], rev. 11. srpen 2010, [vid. 2020-11-18].
Dostupné z: http://www.net-snmp.org/wiki/index.php/TUT:Writing_a_Dynamically_Loadable_Object

[2] TUT:mib2c General Overview. [online], rev. 19. duben 2012, [vid. 2020-11-18].
Dostupné z: http://net-snmp.sourceforge.net/wiki/index.php/TUT:mib2c_General_Overview

[3] netSnmpTutorialMIB.c. [online], rev. 2014, [vid. 20120-11-18].
Dostupné z: https://code.rdkcentral.com/r/plugins/gitiles/rdkb/components/opensource/ccsp/CcspSnmpPa/+/6f60f692a4ccb3286beebeb5cd3bd5b2e7ce087e/source/SnmpPlugin/netSnmpTutorialMIB.c

[4] Net-SNMP Tutorial – MIB Module. [online], rev. 22. květen 2020, [vid. 2020-11-18].
Dostupné z: http://www.net-snmp.org/tutorial/tutorial-5/toolkit/mib_module/

[5] Agent Architecture. [online], rev. 24. listopad 2015, [vid. 2020-11-18].
Dostupné z: http://www.net-snmp.org/wiki/index.php/Agent_Architecture

[6] KLYNE, G.; NEWMAN, C.: Date and Time on the Internet: Timestamps. [online], rev. July 2002, [vid. 2020-11-18].
Dostupné z: https://tools.ietf.org/html/rfc3339

[7] MATOUŠEK, P.: *Síťové služby a jejich architektura*. Publishing house of Brno University of Technology VUTIUM, 2014, ISBN 978-80-214-3766-1, 341-351 s.
Dostupné z: https://www.fit.vut.cz/research/publication/10567