
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**


# ROZŠÍRENIE SNMP AGENTA

**PROJEKT DO PREDMETU ISA**


**AUTOR PRÁCE** **LUKÁŠ TKÁČ, 3BIT**

**ZADÁVATEĽ ZADANIA** **Ing. MICHAL KOUTENSKÝ**



**BRNO 2020**

# Obsah

**1 Úvod** **2**

**2 Teoretická časť** **3**
2.1 User Datagram Protocol(UDP) . . . . . 3
2.2 Architektúra SNMP . . . . . 4
2.3 Protokol SNMP . . . . . 8

**3 Praktická časť** **10**
3.1 1. KROK: Vytvorenie MIB definície. . . . . 10
3.2 2. KROK: Vygenerovanie kostry pomocou utility mib2c. . . . . 11
3.3 3. KROK: Implementácia funkcionalít zdrojových súborov agenta a vytvorenie dynamicky načítateľného súboru do agenta. . . . . 12
3.4 4. KROK: Postup načítania modulu s objetkami a príklady získania informácií. 13

**4 Záver** **16**

**Literatúra** **17**

# Kapitola 1

# Úvod

V tomto projekte sa zaoberám programovaním sieťovej služby do predmetu sieťových aplikácií a správy sietí. Presnejším zameraním tohoto zadania je: "**Rožšírenie SNMP agenta**". Autorom tohoto zadania, je pán **Ing. Michal Koutenský**, kde mu zároveň za toto zadanie pri tejto príležitosti ďakujem. Výsledným riešením bude funkčný **MIB** modul a dynamicky načítateľné rozšírenie SNMP agenta **net-snmp**. Štyri SNMP objekty pod vlastnými **OID** číslami a fungujúcim dotazovaním pomocou utility **snmpget**.

**Nasledujúci dokument resp. manuál je rozdelení do troch nasledujúcich kapitol:**

- Teoretická časť
- Praktická časť
- Literatúra

Pri teoretickej časti, sa pozriem na architektúru prenosového protokolu SNMP a protokol SNMP samostatne. V primeranom rozsahu budem prechádzať prvky architektúry, popis štruktúr, ako sa monitorované objekty identifikujú, ukladajú, dotazujú alebo z čoho sa skladajú. Pri protokole vysvetlím, čo problematika SNMP zahrňuje, z čoho sa skladá. V neposlednom rade, či je ideálnym riešením použiť práve tento protokol a nemali by sme sa porozhliadnuť po inom vhodnejšom riešení.

Pri praktickej časti so zapojením vedomostí z predošlej kapitoly sa pozriem detailnejšie na implementovanie mojej definície MIB modulu a objektov v ňom. Ďalej sa pozriem na aplikáciu net-snmp a jej použitie implementácie SNMP protokolu nasledovne na zdrojové súbory, ktoré využíva agent a následné vytvorenie dynamicky načítatelného binárneho súboru. Budú obsiahnuté taktiež praktické príklady zo získavania informácií zo všetkých implementovaných objektov.

# Kapitola 2

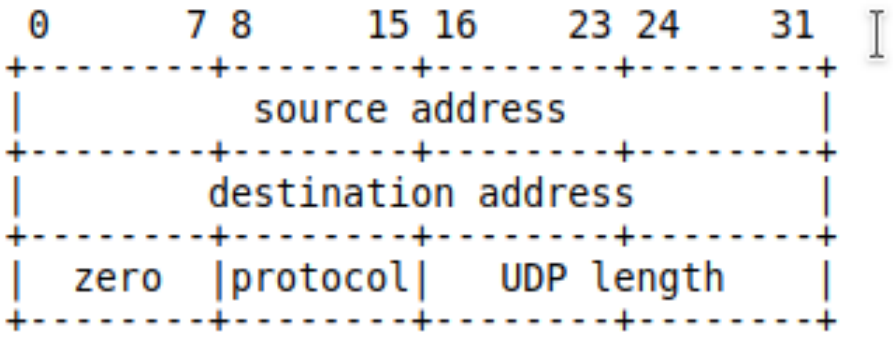
# Teoretická časť

**Nasledujúca kapitola sa bude zaoberať fungovaním služby SNMP na správu siete a to v podkapitolách:**

- User Datagram Protocol(UDP)
- Architektúra SNMP
- Protokol SNMP
- Aplikácia net-snmp

## 2.1 User Datagram Protocol(UDP)

Protokol UDP [5] umožnuje rýchly prenos paketov bez zajisťovania spoľahlivého doručenia. Implementuje tazvané nespojované služby (connection-less). Zdrojový uzol musí sám zaistiť potvdzovanie (lebo je možné, že paket sa stratí) a taktiež doručenie v správnom poradí. **Prenos UDP sa používa prevažne pri aplikáciach prenášajúcich menšie objemy dát v pakete a s dôrazom na rýchlosť doručenia. Medzi také patria napríklad aplikácie na správu a riadenie siete (SNMP, DNS).**

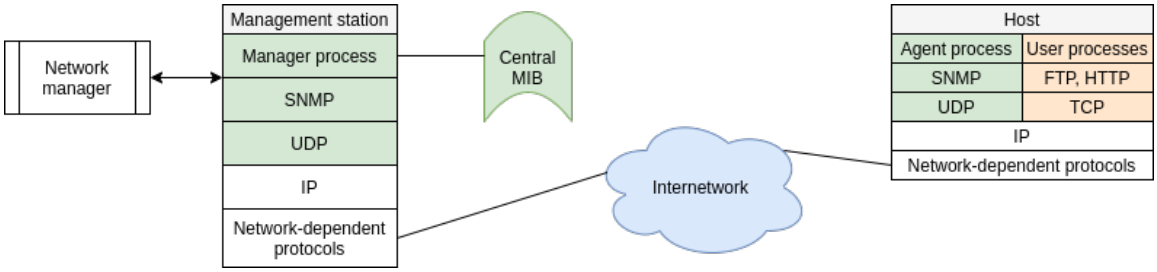



Obr. 2.1: UDP formát [7]

## 2.2 Architektúra SNMP

**Architektúra SNMP** [5] nebola počiatočnou možnosťou pri správe sietí. V raných dobách správy sietí bola nad profilom protokolu TCP/IP založená len na protokole ICMP [8]. S rastúctim počtom uzlov v sieti už nebolo možné kontrolovať stav zariadení len elementárnymi ICMP dotazmi a vtedy sa začalo uvažovať o tom, že by bolo vhodné navrhnúť nový systém pre správu sietí. Prvým pokusom bol protokol SGMP (Single Gateway Monitoring Protocol) [2] z roku 1988 a následne z neho sa vyvinul už spomínaný protokol SNMP (Simple Network Management Protokol). [4]

Architektúra SNMP môže vyzerať nasledovne, viď obrázok 2.2.




Obr. 2.2: Architektúra SNMP

Architektúru SNMP tvoria tieto 4 základné prvky:

### Riadiaca stanica NMS (Network Managment Station)

Tu sa jedná o serverovú aplikácu, ktorú tvoria nástroje na zber dát, na analýzu monitorovaných dát, ukladanie štatistík a prezentáciu siete. Riadiaca stanica môže taktiež nastavovať stav objektov, ktoré sú monitorované podľa požiadavkov, ktoré sú kladené na riadenú sieť.

### Agent na monitorovanom zariadení (Management Agent)

Jedná sa o aktívny proces bežiaci na monitorovanom zariadení, kde pri behu zbiera informácie o prevádzke (napr. kongiguráciu, štatistiky prenosov a bežiacich procesov ...). Veľká časť aktívnych prvkov siete, ktoré sú spravovateľné (server, switch, sonda ...) podporuju SNMP a monitorovací agenti na nich bežia.
**Agent** zasiela objekty obsahujúce monitorované údaje na základe vyžiadania riadiacej stanice. Agent je schopný posielať sám **asynchronné správy typu trap**. Objektom spravovaných SNMP agentom je možné nastavenie prístupu a to napr. **len čítanie (read-only)**, **len zápis (write-only)** alebo **sú umožnené obe operácie (read-write)**.

### Databáza monitorovaných objektov MIB (Management Information Base)

**Databáza MIB** je skupina objektov na monitorovanom zariadený. **Objekt** v databáze je datová štruktúra, ktorá reprezentuje monitorované informácie (napr. za pomoci textového reťazca naformátovaný aktuálny systémový čas). **Monitorovanie siete riadiacou stanicou** prebieha tak, že si vyžiada objekty zo sledovaných zariadení. Riadiace stanice majú schopnosť taktiež niektoré objekty meniť - zapisovať do nich hodnotu napr. zápis 32 bitového čísla so znamienkom reprezentujúcim stavové číslo objektu s nejakým definovaným významom pre dané číslo.

### Prenosový protokol SNMP (Simple Network Management Protocol)

Protokol slúži k predávaniu správ o sledovaných objektoch medzi riadiacou stanicou a agentom. Obsahuje tri základné typy príkazov:

- **get**: riadiaca stanica si vyžiada pomocou tohoto príkazu hodnoty daného objektu od agenta
- **set**: riadiaca stanica nastaví pomocou tohoto príkazu hodnoty daného objektu na agentovi
- **trap**: agent oznamuje pomocou tohoto príkazu riadiacej stanici, že nastala nejaká dôležitá udalosť na monitorovanom zariadení

### Identifikátori objektov a popis ich štruktúry

V štandarte RFC 1155 [9] je popísaná štruktúra a identifikácia monitorovaných objektov vyjadrených pomocou jazyka SMI (Structure Management Information). Tento jazyk definuje pravidlá ako majú byť vytvárané SNMP objekty popísané notáciou ASN (Abstract Syntax Notation One) [10]. Notácia jazyka ASN.1 je používaná pre popísanie abstraktných datových štruktúr a datových typov, ktoré si vymieňajú komunikačné protokoly. Použitie tejto notácie umožnuje komunikovať zariadeniam od rôznych výrobcov, s rôznou architektúrou a s rôznorodými operačnými systémami, ktoré si môžu tieto štruktúry interpretovať rôznymi spôsobmi. ASN.1 zabezpečuje to, že hodnoty z rôznych monitorovaných zariadení sú riadiacej stanici interpretované rovnakým spôsobom v jednotnej notácií.

Na základe tejto skutočnosti sa SNMP objekty popisujú prostredníctvom ASN.1. Každý takto opisovaný objekt má svoje meno, syntax a kódovanie, ktoré je využité pri prenose po sieti.

- **Meno**: objektu je dané unikátnym identifikátorom **OID**
- **Syntax**: objektu definuje prepojenie abstraktného datového typu s objektom. Príklad používaných datových typov ASN.1 (viď obrázok 2.3 a obrázok 2.4.).
- **Kódovanie**: určuje to ako sú inštancie objektu reprezentované pri prenose po sieti. Jazyk SMI používá pre kódovanie štandard BER (Basic Encoding Rules). [3]

| Základné datové typy ASN.1 | |
|---|---|
| Datový typ | Popis |
| INTEGER | 32b celé číslo definované v ASN.1 |
| OCTET STRING | bin/text reťazec definovaný v ASN.1 |
| OBJECT IDENTIFIER | priradedené ASN.1 |
| Integer32 | 32b celé číslo |
| Unsigned32 | kladné 32b celé číslo |
| IPaddress | 32b IP adresa v sieťovom formáte |
| NetworkAddress | pre reprezentáciu iných typov adries |
| Counter32 | 32b čítač, po pretečení sa nastaví na 0 |
| Counter64 | 64b čítač |
| Gauge32 | 32b čítač, po pretečení do resetu uchová max. hodnotu |
| TimeTicks | čas meraný v stotinách sekundy od danej udalosti |
| Opaque | neinterpretovaný reťazec podľa ASN.1 |

Obr. 2.3: Základné datové typy ASN.1

| Rozšírené datové typy ASN.1 | |
|---|---|
| Datový typ | Popis |
| IMPORTS | položky definované v inom module MIB |
| MODULE-IDENTITY | modul s administratívnymi informáciami (napr. kontakt) |
| SEQUENCE | zoznam základných i rozšírených datových typov ASN.1 |
| OBJECT-TYPE | datový typ, prístup ku objektu, stav, a textový popis objektu |
| MODULE-ENTITY | umožnuje spojiť príbuzné obejkty do jedného modelu |
| NOTIFICATION-TYPE | informácie týkajúce sa správ SNMPv2-Trap a InformationRequest |
| MODULE-COMPLIANCE | definuje množiny monitorovaných objektov v module, ktoré musí agent implementovať |
| AGENT-CAPABILITIES | špecifikuje podporované možnosti agenta SNMP |

Obr. 2.4: Rozšírené datové typy ASN.1

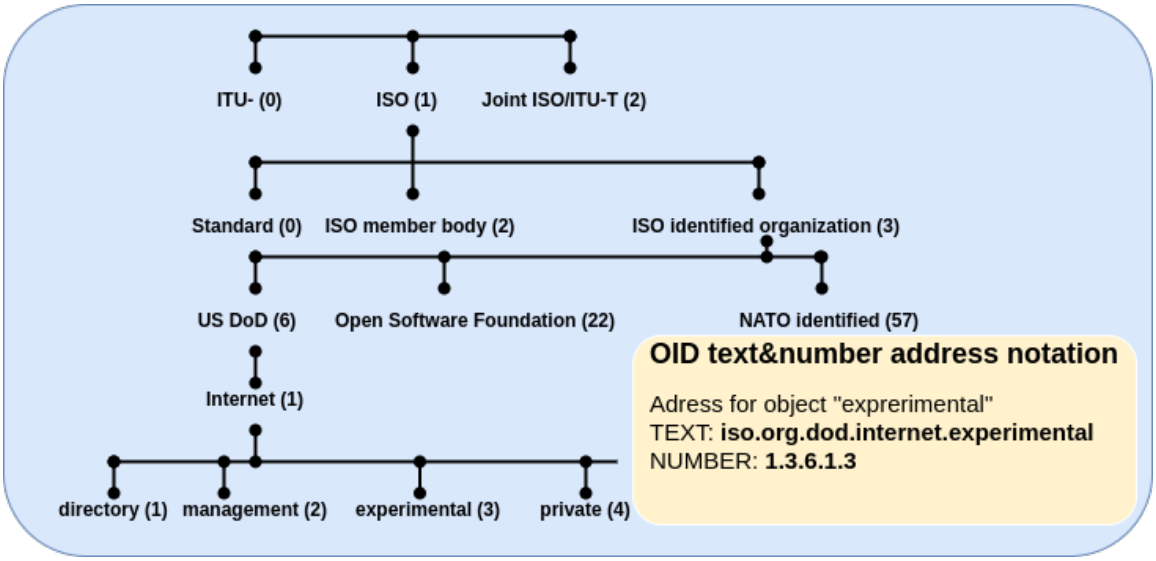
Na základe spomenutých datových typov môžeme vytvárať vlastné objekty s položkami, ktoré chceme o objekte sledovať. Obecná **definícia objektu** vypadá nasledovne:

```
<name> OBJECT-TYPE
    SYNTAX <datatype>
    MAX-ACCESS <read-only|read-write|write-only|not-accessible|accesible-for-notify>
    STATUS <mandatory, optional, obsolete, current, deprecated>
    DESCRIPTION "Popis tohoto vytvorenho objektu."
    ::= {<Unique OID>}
```

Ako je zjavné, tak na definíciu objektu je potrebné mať definované minimálne 4 parametre a to tieto:

- **Syntax**: Definuje akého datového typu objekt je. Viď obrázky 2.2 a 2.3.
- **Max-Access**: Popisuje, aké sú prístupové práva pre premennú a jej hodnoty. Či je povolené čítanie alebo i písanie alebo jej prístup není povolený - je nedostupná.
- **Status**: Popisuje aktuálny stav premennej a to či je: povinná, voliteľná, zastaralá, platná alebo neplatná.
- **Description**: Popisuje použitie objektu v znakovej textovej sade ASCII

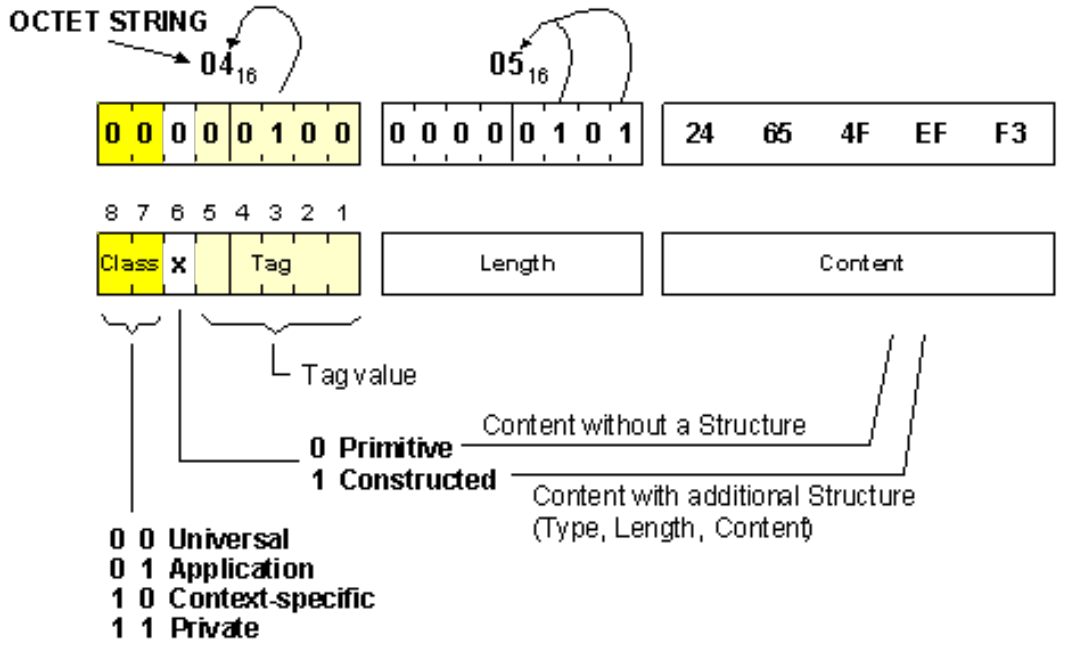
Ako identifikovať alebo sa odkazovať na tieto objekty je popísané štandardom RFC 1155 [9]. Identifikácia je sprostredkovaná za pomoci unikátneho identifikátora **OID**, ktorý môže mať textovú alebo číselnú podobu. Štruktúra je situovaná hiearchicky do určitej formy stromu viď obrázok 2.5, na ktorom následne ukázané ako vyzerá číselná a textová notácia OID.




Obr. 2.5: Identifikácia objektov v MIB

Hodnota objektu protokolu SNMP [4] sa pri prenose po sieti zapisuje do binárneho formátu pomocou kódovania BER (Basic Encoding Rules) [3], kde reprezentácia prenášaných dát je v oktetoch. Zápis je riešení pomocou štruktúry Type-Lenght-Value(TLV), ktorá je vhodná pre prenos vlastnej hodnoty a ku identifikácií prenášaného obsahu.

Tento princíp je efektívny pre stanice, ktoré príjmajú dáta. Vedia koľko pamäte si vyhradiť a to pretože vďaka tejto štruktúre, v ktorej sú dáta poslané, majú k dispozícií akého typu dáta prišli.




Obr. 2.6: ASN.1 BER encoding [6]

## 2.3 Protokol SNMP

Jedná sa o aplikačný protokol nad TCP/IP prenášaný transportným protokolom UDP (User Datagram Protocol). Jeho hlavnou úlohou v archutektúre SNMP je komunikácia medzi riadiacou stanicou a agentom bežiacim na monitorovanom zariadení. Keďže SNMP protokol je prenášaný UDP protokolom, tak z toho vyplýva, že to je protokol typu request-response. UDP je nestavový protokol, čo znamená, že odpovede sú asynchrónne (nezávislé) a nepotvrdené.

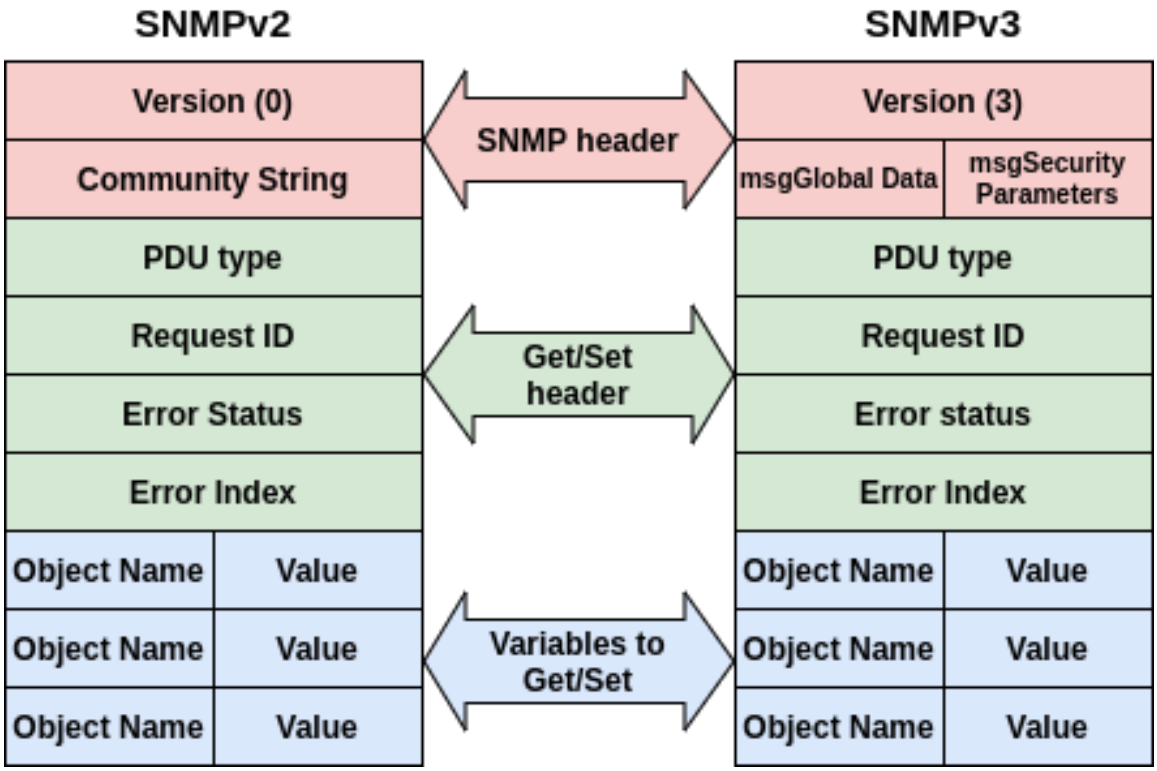
### Mali by sme siahnuť po inej službe spravovania a monitorovania zariadení sieti?

Z predchádzajúceho textu vyplýva, že ak riadiaca stanica pri komunikácii s agentom posiela žiadosti o komunikáciu resp. vyzýva ho k nej (tzv. **pooling**) aby zistil alebo zmenil nejaké hodnoty premenných v objekte na zariadení sledovanom agentom. Kvôli použitiu UDP nevieme, či požiadavka vlastne prišla. A to za predpokladu, že sme nemenili komunikovanie

služby za pomoci použitia pokročilých algoritmov pre komunikáciu v UDP (algoritmus práce konkurentného alebo iteratívneho UDP serveru).

Čo ak by posielal agent mimoriadnu správu typu **trap**, ktorú by riadiaca stanica nedostala (pretože UDP) ? Je to jednou z kritických trhlín tohoto protokolu. Ďalšou trhlinou je bezpečnosť, ktorá až do roku 2002 a verzie protokolu SNMPv3 [1], bola nepoznaná týmto protokolom. Dáta sa dali odposlúchať alebo dokonca si ich mohol ktokoľvek vyžiadať. Nepomohla ani prvotná záplata bezpečnosti vo forme **community stringu**, keďže nebolo naň použité žiadne šifrovanie na anonymizáciu packetu pred potencionálnym útočníkom.

Protokol SNMPv3 definuje bezpečnostný mechanizmus, ktorý zabezpečuje integritu dát a autentizáciu odosielateľa. Začali sa k tomu používať algortimy **HMAC-MD5** a **HMAC-SHA**, kde SNMP protokol nezaisťuje šifrovanie týmito algoritmami. Šifra je vložená do SNMP hlavičky už zašifrovaná.




Obr. 2.7: Formát paketov SNMP protokolu [6]

# Kapitola 3

# Praktická časť

V tejto kapitole bude opísaná praktická časť projektu. Popísaná bude implementácia jednotlivých častí, ďalej ich čiastočný popis, postup zbudovania projektu a následne použitie programu s požadovanou utilitou **snmpget**. Zdokumentovanie objektov a to hlavne toho požadovaného **.1.3.6.1.3.22.4.0**. Vybral som si variantu, kde dávam ešte na koniec každého objektu **.0** a môj objekt (moja štvrtá premenná) bude monitorovať **množstvo RAM**.

**Objekty sú teda vyzývané (pooling) pod nasledovnými OID:**

- Read-only string s mojím loginom (**.1.3.6.1.3.22.1.0**)
- Read-only string, ktorý vracia aktuálny čas naformátovaný podľa RFC 3339 (**.1.3.6.1.3.22.2.0**)
- Read/write Int32 (**.1.3.6.1.3.22.3.0**)
- Read-only string, ktorý informuje o množstve RAM v systéme) (**.1.3.6.1.3.22.4.0**)

V nasledujúcich krokoch bude popísaný postup od počiatku až po finálny stav projektu.

## 3.1 1. KROK: Vytvorenie MIB definície.

MIB definícia je vytvorená v súbore s formátom txt. Za pomoci jazyka SMI s notáciou ASN.1. (oboje spomenuté v teoretickej časti dokumentácie). v definíciach MIB-u som začal pomocou **IMPORTS**, kde som si naimportoval položky, ktoré sú definované v iných moduloch MIB. Definoval som si identitu modulu (**MODULE-IDENTITY**), doplnil kontaktné informácie a informácie ohľadne modulu. Podstrom implementovaných objektov je pod objektom **experimental a to pod číslom 22**.

Definícia identity modulu, ktorá je implementovaná s menom **xtkaclMIB** je nasledovná:

```
xtkaclMIB MODULE-IDENTITY
    LAST-UPDATED "202011140000Z" -- 14 Nov 2020, midnight
    ORGANIZATION "vut-brno"
    CONTACT-INFO " created by: Lukas Tkac
                 email: xtkacl00@stud.fit.vutbr.cz"
    DESCRIPTION "A simple mib for demonstrate purposes
    in Network Applications and Network Administration univeristy subject."
    REVISION "202011140000Z"
    DESCRIPTION " The latest version of this MIB module."
        ::= { experimental 22 }
```

Mnou vybraný objekt ku implementovaniu je objekt, ktorý informuje o množstve RAM na vybranom zariadení (**OID: .1.3.6.1.3.22.4**). Implementácia:

```
myRAM OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    " This is an object which gives number value of system RAM. (Random Access Memory)
    DEFVAL { 0 }
        ::= { xtkaclMIB 4 }
```

**Objekt myRAM:**

- Je datového typu DisplayString (OCTET STRING).
- Je prístupný maximálne na čítanie.
- Stav premennej je, že je platná
- Popis objektu.

## 3.2 2. KROK: Vygenerovanie kostry pomocou utility mib2c.

Po preskúmaní manuálových stránok net-snmp som narazil na túto utilitu, ktorá podľa zvolenej konfigurácie z ponúkaných vygeneruje kostru so zdrojovými súbormi agenta (mib2c utilita). Pre generovanie som použil configuračný súbor mib2c.scalar.conf. Za použitia ktorého je pre každý nadefinovaný objekt z MIB súború vytvorený handler pre každý skalárny objekt.

Postup bol nasledovný:
1. Vložil som MIB s definíciami(XTKACL-MIB.txt) do mibs/ priečinka (/usr/share/snmp-d/mibs/) snmp deamona SNMP agenta a načítal MIB zadaním do terminálu nasledovného: **export ="+XTKACL-MIB"**

2. Vygeneroval som kostru zdrojových súborov agenta zadaním do terminálu nasledovného: **mib2c -c mib2c.scalar.conf xtkaclMIB**. Vygenerované zdrojové súbory sú: **xtkacl-MIB.c** a **xtkaclMIB.h**. Do nich som doplnil požadovanú funkcionalitu v ďalšom kroku.

## 3.3 3. KROK: Implementácia funkcionalít zdrojových súborov agenta a vytvorenie dynamicky načítateľného súboru do agenta.

Väčšina implementácie je viditeľná v zdrojových súboroch, takže vyberiem jeden z objektov a jeho zdrojový kód. Bude to objekt s **OID .1.3.6.1.3.22.4** a to objekt **myRAM**(Automaticky vygenerované komentáre som zmazal kvôli miestu v dokumentácii - v zdrojových súboroch sú ponechané). Zdrojový kód je takýto a funkcionalita popísaná v komentároch na riadkoch kódu. Na zistenie tejto systémovej informácie o RAM pamäti bola použitá knižnica sysinfo.

```
int
handle_myRAM(netsnmp_mib_handler *handler,
                        netsnmp_handler_registration *reginfo,
                        netsnmp_agent_request_info *reqinfo,
                        netsnmp_request_info *requests)
{
    struct sysinfo system_info; // declare system info data structure
    sysinfo(&system_info); // determine system info into sysinfo data stucture
    char sys_buff[30]; // char buffer with fixed size for result value
    memset(sys_buff, 0, sizeof(sys_buff)); // clear buffer

    uint64_t ram_size_mb = SIZE_IN_MB; // MAGIC number for B->MB conversion
    // compute total ram size in bytes to size in megabytes
    uint64_t ram_in_mb = system_info.totalram / ram_size_mb;
    // push value into buffer with unit name of value
    sprintf(sys_buff, "%"PRIu64" MB", ram_in_mb);

    /**
     * Via OCTET STRING are printable characters as '/0',
     * so i decided that i am going to count text size without them). :)
     */
    int i = 0;
    int text_size = 0;
    while(sys_buff[i] != '\0') {
        text_size++;
        i++;
    }

    switch(reqinfo->mode) {

        case MODE_GET:
            snmp_set_var_typed_value(requests->requestvb, ASN_OCTET_STR,
            &sys_buff, sizeof(char) * text_size);
            break;
        default:
            /* we should never get here, so this is a really bad error */
            snmp_log(LOG_ERR, "unknown mode (%d) in handle_myRAM\n", reqinfo->mode );
```

```
            return SNMP_ERR_GENERR;
    }

    return SNMP_ERR_NOERROR;
}
```

Funkcia v **case GET** je popísaná na tomto mieste v doxygene aplikácie net-snmp na tomto odkaze.

Poznámka ku objektu(**myNumGetSet**), ktorý je prístupný na čítanie aj zápis (operácie get aj set), kde tu presne čítam hodnotu, ktorá je uložená do globálnej štruktúry s premennou rovnakého datového typu ako datový typ objektu. Takže po požiadavke get dostaneme naposledy vloženú hodnotu. Platí to dokým neuvolníme modul s týmto objektom. Potom je nastavená default value ako aj na počiatku a to je 0.

### Vytvorenie dynamicky načítateľného súboru do bežiaceho agenta

Na to aby sme mohli dynamicky načítavať je vhodné aby sme si vytvorili kompiláciou a zlinkovaním zdrojových súborou agenta tzv. shared object file (.so). Ak používame gcc prekladač väčšinou je potrebné aby sme zahrnuli pri tejto operácií flag **-fPIC -shared**.

Na štúdium ako na to som použil zdroje a informácie z lokality patriacej net-snmp s návodom z tohoto odkazu.

Pre túto úlohu je použitý nasledovný makefile.

```
CC=gcc
CFLAGS=-I. `net-snmp-config --cflags`

# shared library flags (assumes gcc)
DLFLAGS=-fPIC -shared

xtkaclMIB.so: xtkaclMIB.c Makefile
	$(CC) $(CFLAGS) $(DLFLAGS) -c -o xtkaclMIB.o xtkaclMIB.c
	$(CC) $(CFLAGS) $(DLFLAGS) -o xtkaclMIB.so xtkaclMIB.o
```

## 3.4 4. KROK: Postup načítania modulu s objetkami a príklady získania informácií.

Použité názvy modulu, objektov, MIB definície, korešpondujú s mojimi názvami, ktoré som odovzdal.

1. V prvom okne terminálu si zapneme snmpd (deamon pre SNMPD agenta) aj s ladiacimi flagmi, kde vďaka tomu vidíme dynamického tvorcu modulu komunikovať s naším modulom:

```
sudo snmpd -f -L -DxtkaclMIB,dlmod
```

2.Prvé okno necháme bežať a otvoríme si druhé okno terminálu, kde si pustíme snmpset na vytvorenie nového riadku v dlmod tabuľke:

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

3. Nastavíme vlastnosti riadku a to tak aby ukazovali na náš nový objekt a dáme mu meno (POZOR: zadávajte ABSOLÚTNU CESTU ku .so súboru !!!):

```
snmpset localhost UCD-DLMOD-MIB::dlmodName.1 s "xtkaclMIB"
UCD-DLMOD-MIB::dlmodPath.1 s "/path/to/xtkaclMIB.so"
```

4. Načítame shared objekt do bežiaceho agenta:

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

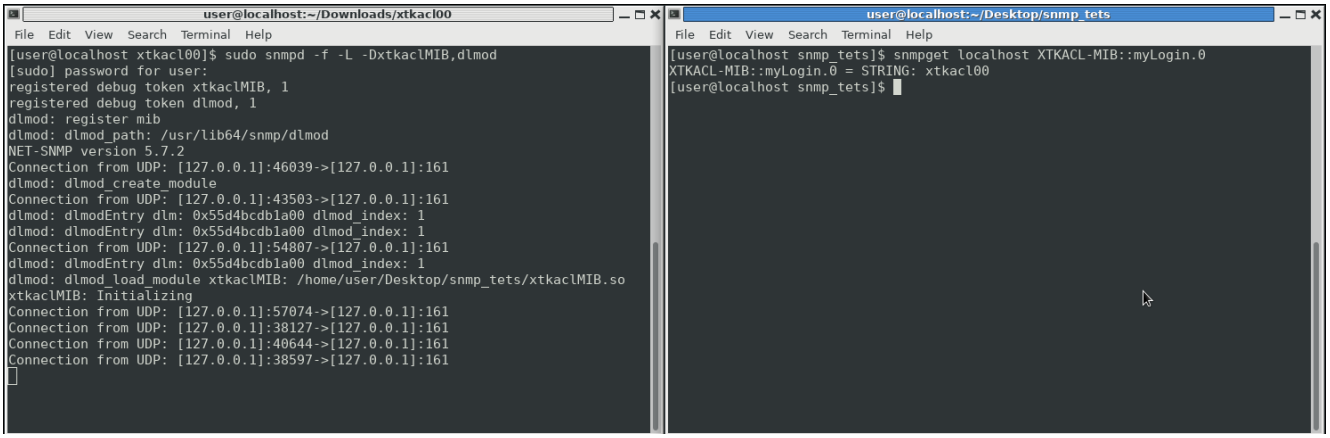
5. Ak všetky kroky sa podarili, tak je možné získavať informácie pomocou utility **snmpget** a to takto napr. na príklade môjho objektu myLogin:

```
snmpget localhost XTKACL-MIB::myLogin.0
```

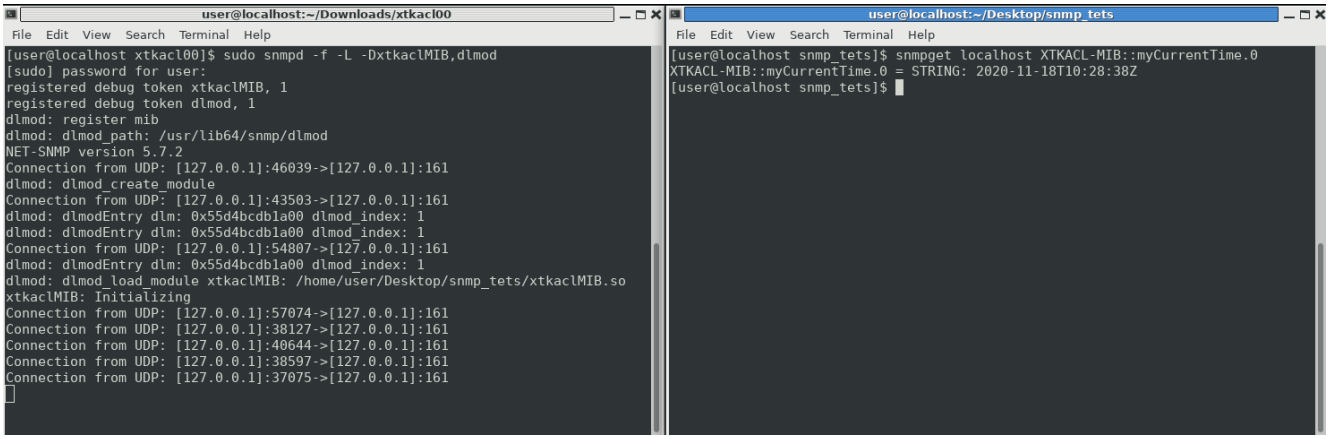
Očakávaná odpoveď: **XTKACL-MIB::myLogin.0 = STRING: xtkacl00**

**Vizuálna ukážka získania informácií zo všetkých definovaných 4 objektov:**

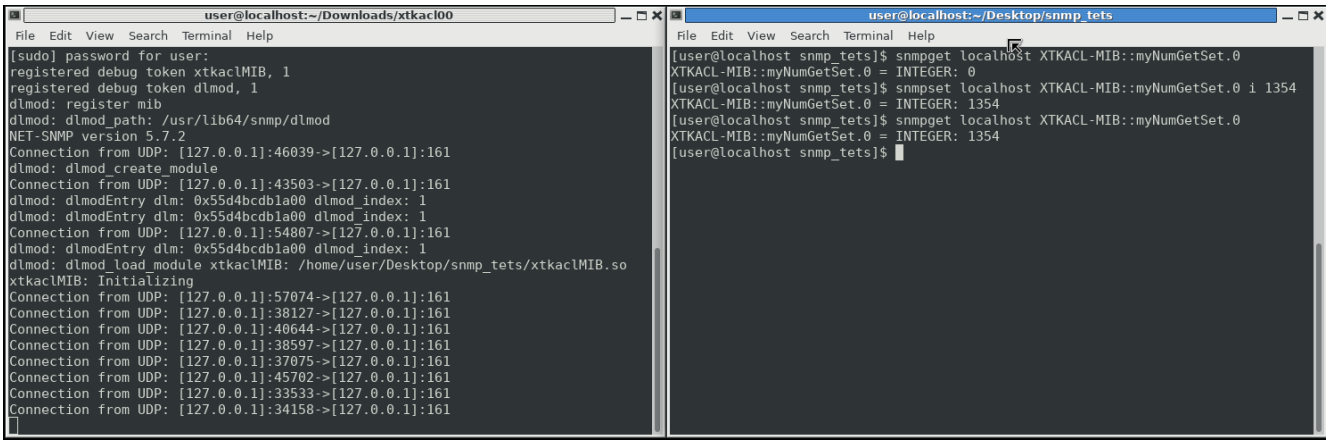
- **myLogin**: uchováva pre čítanie informáciu s mojím loginom xtkacl00 viď obrázok 3.1.
- **myCurrentTime**: uchováva pre čítanie aktuálny čas podľa GMT časového pásma vo formáte podľa RFC 3339 viď obrázok 3.2.
- **myNumGetSet**: uchováva pre čítanie a zápis 32 bitové číslo datového typu Integer viď obrázok 3.3.
- **myRAM**: uchováva pre čítanie množstvo RAM v textovej forme viď obrázok 3.4.



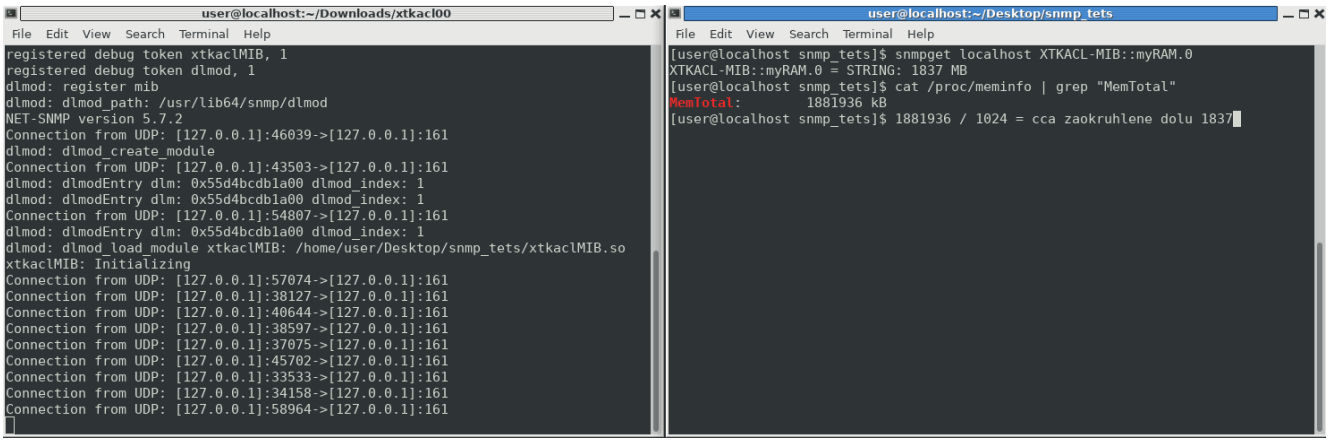

Obr. 3.1: Získavanie informácie od objektu myLogin.




Obr. 3.2: Získavanie informácie od objektu myCurrentTime.




Obr. 3.3: Získavanie informácie od objektu myNumGetSet.




Obr. 3.4: Získavanie informácie od objektu myRAM.

# Kapitola 4

# Záver

Počas tohoto projektu som zistil mnoho okolo problematiky, bezpečnosti, výhod a nevýhod architektúry SNMP a ako vlastne veci tu fungujú. Za túto skúsenosť som rád. Chcel by som vvyužiť záver k poďakovaniu pánovi garantovi Petrovi Matouškovi, že mi jeho publikácia bola takým vodítkom kadiaľ teoretickiú časť uberať a dobrým detailným zdrojom informácii. Taktiež by som i Vám chcel poďakovať pán inžinier Koutenský za to celé zadanie, za poskytnuté materiály, ktoré boli priložené ako zdroje ku zadaniu a boli veľmi nápomocné pri praktickom vypracovaní projektu.

Ďakujem !

# Literatúra


[1] Case, J., Mundy, R., Partain, D. a Stewart, B. *Introduction and Applicability Statements for Internet Standard Management Framework* [online]. RFC 3410, december 2002 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc3410.

[2] Davin, J., J.Case, Fedor, M. a Schoffstall, M. *A Simple Gateway Monitoring Protocol* [online]. RFC 1028, november 1987 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc1028.

[3] ITU T, T. S. S. O. *X.690 : Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)* [online]. X.690, august 2015 [cit. 2020-11-17]. Dostupné z: https://www.itu.int/rec/T-REC-X.690-201508-I/en.

[4] J.Case, Fedor, M., Schoffstall, M. a Davin, J. *A Simple Network Management Protocol* [online]. RFC 1067, august 1988 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc1067.

[5] Matoušek, P. *Síťové aplikace a jejich architektura*. 1. vyd. Brno: VUTIUM, 2014. ISBN 978-80-214-3766-1.

[6] NettedAutomation a JohnBlack'01. *The basis of the ASN.1 Basic Encoding Rules (BER, ISO 8825)* [online]. November 2000 [cit. 2020-11-17]. Dostupné z: https://www.nettedautomation.com/standardization/iso/tc184/sc5/wg2/mms_intro/intro6.html.

[7] Postel, J. *User Datagram Protocol* [online]. ISI, august 1980 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc768.

[8] Postel, J. *Internet Control Message Protocol* [online]. ISI, apríl 1981 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc777.

[9] Rose, M. a McCloghrie, K. *Structure and Identification of Management Information for TCP/IP-based Internets* [online]. RFC 1155, máj 1990 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc1155.

[10] Wallace, C. a Gardiner, C. *ASN.1 Translation* [online]. RFC 6025, október 2010 [cit. 2020-11-17]. Dostupné z: https://tools.ietf.org/html/rfc6025.