VŠB – Technická univerzita Ostrava

Fakulta elektroniky a informatiky

Směrované a přepínané sítě

Semestrální projekt:

TCL skriptování pod prvky Cisco

LS 2007

Aleš Ogrocký,
Svatopluk Uličný

Obsah

1. Úvod

TCL neboli Tool Command Language je skriptovací jazyk, jehož vznik se datuje koncem roku 1988. Od počátku byl jazyk vyvíjen tak, aby byl zároveň jednoduchý, rozšiřitelný, multi-platformní a snadno implementovatelný do různých aplikací. Právě jedno z využití, kterým se budeme v tomto textu zabývat, je skriptování v prostředí prvků (směrovače a přepínače) společnosti Cisco. Nejen k často opakovaným sekvencím příkazů se jistě možnost vytvářet a spouštět skripty může hodit. Další výhodu TCL skriptování může být třeba u formátování výstupů, kdy administrátora často jen zajímá určitá informace, zatímco je zavalen kompletním výpisem. Dále jistě není na škodu komunikace s okolím, kdy například můžeme zpracovat informace z SNMP. Samozřejmě i obsluha událostí najde rovněž široké uplatnění.

2. Požadavky a vlastnosti TCL skriptování

Předtím, než se dostaneme k požadavkům, zmíníme, že prvky Cisco pracují se specifikací jazyka TCL (verze 8.3.4 a vyšší) prostřednictvím Cisco IOS command-line interface (CLI).
Abychom mohli výše uvedené možnosti používat, potřebujeme následující:

- Prvek s operačním systémem Cisco IOS 12.3(2)T a vyšší (částečná podpora od verze 12.2(25)S). V zatím poslední verzi 12.3(7)T je doplněna podpora pro práci SNMP MIB objekty.
- TCL skripty se spouštějí prostřednictvím konfiguračního režimu TCL („**tclsh**“), na který se přistupuje z privilegovaného režimu.
- Možnost spuštění TCL skriptu z Embedded Event Manager (EEM) na základě událostí (čas, příkaz).

TCL skriptování má i určité vlastnosti, na které je vhodné upozornit:

- Psaní příkazů a podpříkazů v konfiguračním režimu (Cisco IOS configuration commands) v TCL skriptech musí být v uvozovkách a na jednom řádku.
- Chyby ve skriptech mohou vést k nekonečné smyčce a „k zamrznutí“. Z vlastní zkušenosti doporučujeme testovat skripty v neostrých podmínkách.

3. Popis možností TCL skriptování

Cílem této práce není naučit jazyk TCL (více o samotném TCL zde http://www.tcl.tk/), ale popsat, jak se s tímto jazykem pracuje v prostředí prvků Cisco. Zde je možno tvořit skripty přes klasické rozhraní (CLI), avšak tento způsob pravděpodobně nebude příliš vhodný pro tvorbu delších skriptů. Místo toho je vhodné použít libovolný TFTP server (například FREE TFTP Server – http://solarwinds.net/downloads/index.aspx). Skripty jsou pak uloženy na

TFTP serveru a dají se snadno editovat, což v prvním případě neplatí. Prvek si pak skript/y jednoduše stáhne a případně uloží do NVRAM či Flash paměti (prostřednictvím příkazu „**copy**“).

Co se týče psaní skriptů, doporučujeme používat komentáře, které jsou uvozeny znakem „#”. Běžně používaným příkazem TCL jazyka „**puts** "*řetězec*"“ vypíšeme na výstup zadaný řetězec. Při debugování si příkazem „puts“ můžeme často pomoci, jelikož interpret TCL poskytuje základní informace o chybě, avšak například číslo řádku chyby již ne.

Nyní si popíšeme rozšiřující příkazy Cisco IOS Custom TCL Command Extensions, které jsou doimplementovány v TCL interpretu a tvoří doplněk jazyka TCL. Zobrazení informací o syntaktických chybách lze aktivovat pomocí příkazu „**log_user** *číslo*“, kde číslo 0 znamená potlačení a číslo 1 zobrazení chybových zpráv. Dalším příkazem, který zapíše řetězec na STDIN prvku je příkaz „**typeahead** "*řetězec*"“. Následující dva příkazy slouží k provedení Cisco IOS příkazů. Jsou to konkrétně „**exec** "*příkaz*"“ a **„ios_config** "*příkaz*" "*podpříkaz*"“. První slouží k provedení Cisco IOS příkazů v privilegovaném režimu a druhý v konfiguračním režimu. Skutečné použití je vidět v následujícím odstavci a kompletní ukázkové příklady včetně výstupů jsou uvedeny v kapitole 4.

3.1 Jak spustit skript (způsob přes CLI)

Nyní si ukážeme jak postupovat při vytvoření/spuštění TCL skriptu prostřednictvím CLI. Pro přehlednost jsou následující kroky očíslovány. Na dalším řádku je kurzívou uveden konkrétní příkaz. Volitelné kroky 2–5 je možné přeskočit, ukazují pokročilé možnosti.

1. Vstup do privilegovaného režimu
   ```
   Router> enable
   ```
2. (volitelné) Vstup do (globálního) konfiguračního režimu
   ```
   Router# configure terminal
   ```
3. (volitelné) Specifikace defaultního umístění skriptů
   ```
   Router(config)# scripting tcl encdir tftp://10.18.117.23/enctcl/
   ```
4. (volitelné) Specifikace, který skript má být spuštěn při inicializaci
   ```
   Router(config)# scripting tcl init
   ftp://user:password@172.17.40.3/tclscript/initfiles3.tcl
   ```
5. (volitelné) Návrat do privilegovaného režimu
   ```
   Router(config)# exit
   ```
6. Aktivace TCL konfiguračního režimu (spuštění TCL interpreta)
   ```
   Router# tclsh
   ```
7. Příkazy TCL skriptu, možnosti:

- Pro příkazy v konfiguračním režimu Cisco IOS, (podpříkaz na stejném řádku!)

  ```
  Router(tcl)# ios_config "interface Ethernet 2/0" "no keepalive"
  ```

- Pro příkazy v priviligovaném režimu Cisco IOS

  ```
  Router(tcl)# exec "show interfaces"
  ```

- Příkazy pro TCL interpret (náš kód)

  ```
  Router(tcl)# puts "Hello, world\n"
  ```

8. Ukončení TCL konfiguračního režimu (TCL interpretu), návrat do privilegovaného režimu

   ```
   Router(tcl)# exit
   ```

3.2 Jak spustit skript (TFTP a další způsoby)

Lepší alternativou skriptování oproti CLI je předem si připravit zdrojový kód skriptu v textovém editoru a pak jej nahrát a spustit na daném prvku. Zde podotkněme, že v jeden čas můžeme spustit více skriptů (interpret TCL a TFTP server je spojen přes TTY). Nejsme omezeni jen na TFTP server, můžeme použít i klasický FTP server, či Cisco IOS File System (IFS). Opět použijme číslovaný seznam bodů, kde za každým bodem je kurzívou uveden konkrétní příkaz.

1. Vstup do privilegovaného režimu

   ```
   Router> enable
   ```

2. Aktivace TCL konfiguračního režimu (spuštění TCL interpreta)

   ```
   Router# tclsh
   ```

3. Příkaz pro načtení TCL skriptu, možnosti:

   - TFTP Server

     ```
     Router(tcl)# source tftp://10.0.0.1/tclscript.tcl
     ```

   - Flash Memory

     ```
     Router(tcl)# source slot0:test.tcl
     ```

   - FTP Server

     ```
     Router(tcl)# source
     ftp://user:password@172.17.40.3/tclscript/test.tcl
     ```

4. Ukončení TCL konfiguračního režimu (TCL interpretu), návrat do privilegovaného režimu

```
Router(tcl)# exit
```

3.3 Možnosti SNMP a MIB

Od verze Cisco IOS 12.3(7)T je umožněno pracovat s SNMP a MIB. Nejdříve si připomeneme, co znamenají jednotlivé zkratky:

SNMP (Simple Network Management Protocol): Původně určen pro usnadnění správy sítě, možnosti jeho využití jsou ale podstatně větší a tak stále častěji proniká i do průmyslové automatizace a měřicí techniky.
Protokol SNMP je asynchronní, transakčně orientovaný protokol založený na modelu klient/server. Strana, která posílá požadavky (snmp klient), může být např. jednoduchý snmp browser či složitý NMS (Network Management Systém), na straně zařízení je snmp agent (snmp server), který na požadavky odpovídá.

MIB (Management Information Base): Databáze, která dovoluje jednoznačně identifikovat informace využívané systémem správy. Aby mohl SNMP manager i agent tyto informace získat a předávat, je nutná znalost struktury MIB.
MIB popisuje sadu objektů, které jsou předmětem správy. Spravované zařízení může implementovat jednu nebo více MIB, v závislosti na jeho funkci. Tyto MIB databáze jsou velmi podobné standardním databázím v tom smyslu, že popisují jak strukturu, tak formát dat. MIB jsou napsány podle pravidel Structure of Management Information (SMI), které jsou popsány v dokumentech RFC1155, RFC1212 a RFC1215. V současnosti již existuje i návrh (draft) standardu SMIv2, zpětně kompatibilního s předchozí verzí.
Je tedy datová hierarchická stromová struktura, která odpovídá danému konkrétnímu zařízení a je objektově orientována jako sada SNMP objektů, relací a operací na a mezi objekty.

S objekty budeme pracovat následujícími příkazy, implementovány v TCL:

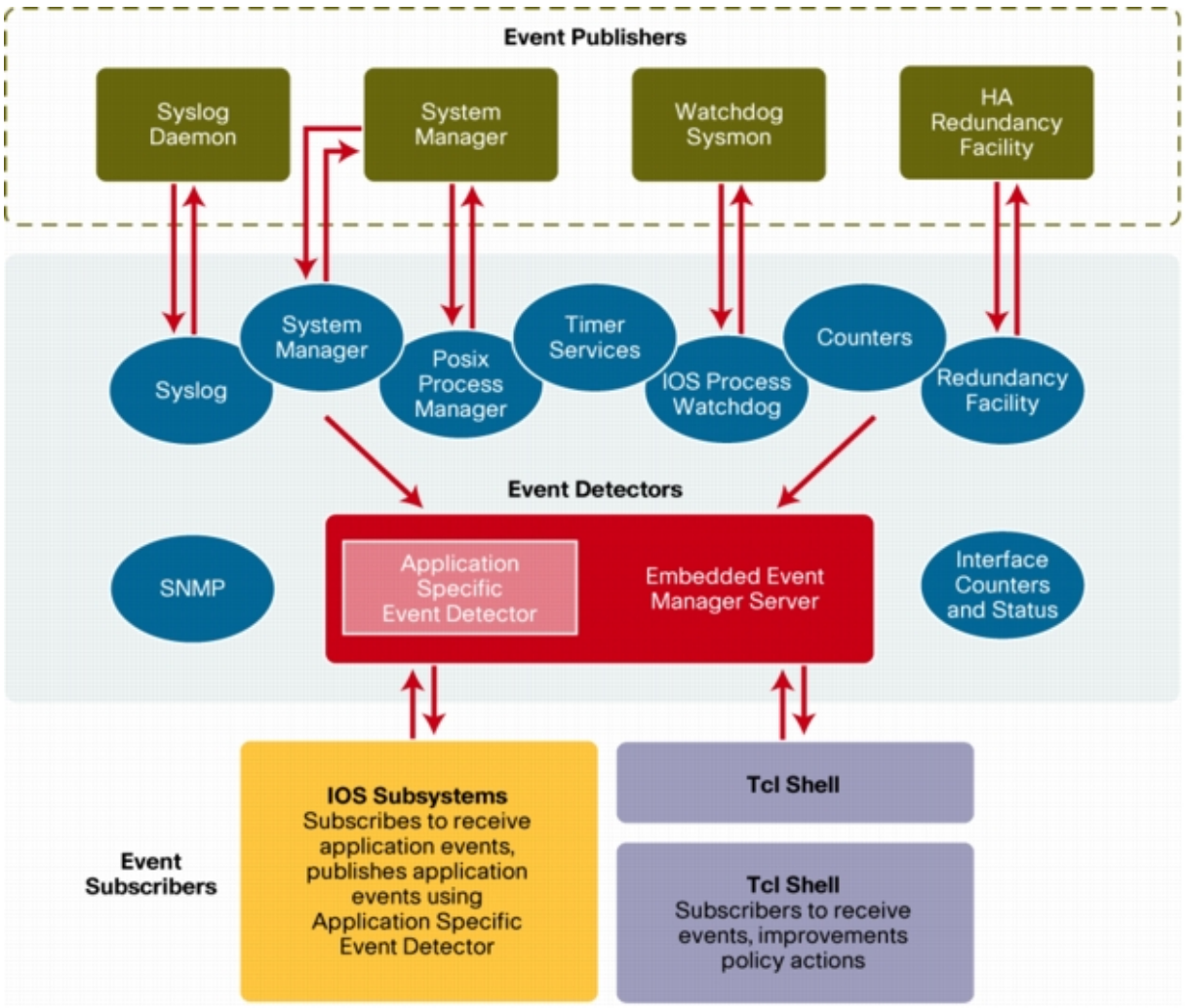
- **snmp_getbulk** *community non-rep reps oid1 oid2...*
  Získání velké části MIB tabulky.
  Parametry:
  community – určuje komunitu, ze které chceme získat objekty
  non-rep – počet objektů, které mohou být získány pomocí get-next operace
  reps – maximální počet opakování get-next operací pro zbývající OID
  oid1 – identifikace objektu
- **snmp_getid** *community*
  Získání šesti základních systémových proměnných SNMP.

community – sysDescr.0, sysObjectID.0, sysUpTime.0, sysContact.0, sysName.0, sysLocation.0

- **snmp_getnext** *community oid1 oid2…*
  Získání množiny individuálních objektů z MIB tabulky
- **snmp_getone** *community oid1 oid2…*
  Získání množiny individuálních objektů z MIB tabulky (vyžadováno plně kvalifikované jméno objektu)
- **snmp_setany** *community oid1 type1 value1 oid2 type2 value2...*
  Modifikování určených objektů
  Parametry: trojice: identifikace objektu, typ, hodnota
  kde typ může být: -i pro integer, -u pro unsigned32, -c pro counter32, -g pro gauge, -o pro octet string, -d pro display string, -ipv4 pro IPv4, -oid pro OID.

3.4 Embedded Event Manager a TCL skripty

Další možnost využití TCL skriptů lze spatřit při zpracování událostí. Embedded Event Manager slouží k obsluze událostí a umožňuje na různé události patřičně reagovat, vytvářet politiky. Od verze EEMv2 jsou zde podporovány dva přístupy. Jedním přístupem je obsluha pomocí CLI apletů, druhým pomocí TCL skriptů. CLI aplet je jednoduchá politika, která definována prostřednictvím CLI příkazů. Naproti tomu politika založená na TCL poskytuje širší možnosti, jelikož je postavena na skriptovacím jazyce. Princip obsluhy událostí pomocí TCL je takový, že nejdříve se politika musí zaregistrovat k určité události. Následně při každém výskytu události EEM vykoná danou politiku. Na následujícím obrázku je znázorněna interakce EEMv2 při obsluze události.



Obrázek 1: Proces reakce na událost (EEMv2)

Z obrázku je patrné jak probíhá zpracování událostí. Jakmile je událost vytvořena je zachycena svým detektorem. Detektor informuje EEM o události a ten vykoná případné zaregistrované politiky. Jak ale nastavit takovéto zpracování událostí? Musíme postupovat podle následujících kroků:

1. Nastavení implicitního adresáře pro skripty politik

```
router2#mkdir ABCCoTclPol
Create directory filename [ABCCoTclPol]?
Created dir disk0:ABCCoTclPol
router2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
router2(config)#event manager directory user policy
disk0:/ABCCoTclPol
router2(config)#^Z
router2#router2#sh event man dir user policy
disk0:/ABCCoTclPol
```

2. Zkopírovaní skriptu/ů politiky

```
router2#copy tftp disk0:
Address or name of remote host []? 10.1.88.9
Source filename []? script.tcl
Destination filename [script.tcl]? ABCCoTclPol/script.tcl
Accessing tftp://10.1.88.9/script.tcl...!
1232 bytes copied in 0.620 secs (1987 bytes/sec)
```

3. Registrace (ověření) politiky

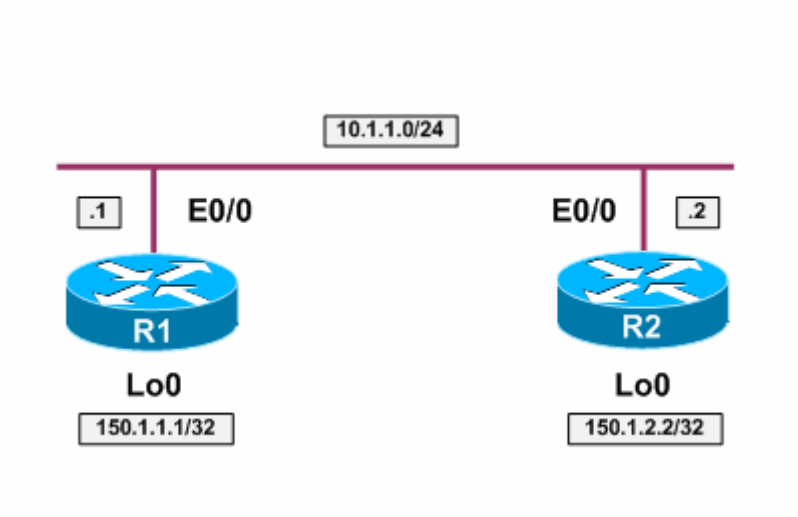
```
router2#conf t
Enter configuration commands, one per line. End with CNTL/Z.
router2(config)#event manager policy script.tcl type user
router2#sh event manager policy registered
No. Type Event Type Trap Time Registered Name
1 user syslog Off Thu Oct30 14:54:17 2003 script.tcl
occurs 1 pattern {%SYS-5-CONFIG_I: Configured}
nice 0 priority normal maxrun 90.000
```

Z uvedených kroků vyplývá, že registrovat lze jen politiky, které jsou uloženy v paměti prvku. Musí být dále specifikován defaultní adresář (**event manager directory user policy**) a do něj nakopírován (**copy**) TCL skript politiky. Registrace (**event manager policy**) se provádí v globálním konfiguračním režimu. Pozorný čtenář si jistě všimne, že politika je sice registrovaná, ale nevíme ke které události je přiřazena. Definice typu události je totiž uvedena na začátku skriptu. Například první řádek politiky *script.tcl* by mohl vypadat takto: **„::cisco::eem::event_register_syslog occurs 1 pattern "\%SYS-5-CONFIG_I: Configured" maxrun_sec 90“**.
Na dalších řádcích by byl uveden zbytek skriptu. Pro jednoduchost předpokládejme, že na druhém řádku je příkaz „**puts** "*Hello world!\n*"“.
V této konkrétní politice se registruje na událost syslogu, kdy očekáváme, že výstup bude odpovídat uvedenému vzoru (pattern). Tedy při každém výskytu „%SYS-5-CONFIG_I: Configured“, EEM spustí *script.tcl* a vypíše „Hello world!“.
Pokud bychom skript editovali je doporučené původní skript „odregistrovat“ (**no event manager policy**) a upravený skript znovu registrovat. Podrobnosti týkajících se různých typů událostí EEM lze nalézt na http://www.cisco.com/en/US/products/ps6815/products_white_paper0900aecd803a4dad.shtml.

4. Ukázkové příklady

4.1 Ping sousedního routeru s použitím RSH

Popis:
Uvedený skript je spouštěn z routeru R1. Nejprve zjistí pomocí RSH všechny IP adresy, které jsou přiřazeny jednotlivým interfacům routeru R2. Poté ověří dostupnost těchto IP adres pomocí příkazu ping.



a) Konfigurace

Router R1 [Rack1R1] - Verze IOS 12.3T

Konfigurace:

```
hostname Rack1R1
no ip domain lookup
interface Loopback0
ip address 150.1.1.1 255.255.255.0
interface Ethernet0/0
ip address 10.1.1.1 255.255.255.0
router rip
version 2
network 10.1.1.0
network 150.1.1.0
no auto-summary
line con 0
line aux 0
line vty 0 4
password cisto
login
```

Router R2 [Rack1R2] - Verze IOS 12.3T

Konfigurace:

```
hostname Rack1R2
ip rcmd rsh-enable
ip rcmd remote-host Rack1R1 10.1.1.1 Rack1R1 enable
no ip domain lookup
interface Loopback0
ip address 150.1.2.2 255.255.255.0
interface Ethernet0/0
ip address 10.1.1.2 255.255.255.0
router rip
version 2
network 10.1.1.0
network 150.1.1.0
no auto-summary
line con 0
line aux 0
line vty 0 4
password cisco
login
```

b) TCL Skript

```
proc ping_all {ip} {
# ping na zadanou IP adresu, jestliže je dostupná pokračuje se,
# určující jsou vykřičníky při úspěšném pingu
# regexp : jestliže výsledek příkazu ping obsahuje "!!" pak je
# podmínka splněna
if {[regexp "(!!)" [exec "ping $ip"]]}{
# nastavíme počítadlo pro počet IP adres
# set : nastaví proměnnou counter na hodnotu 0
set counter 0
exec "term len 0"
# zjistime hostname, rozhraní a IP adresy
# lindex : naplní promennou hostname 2 retězecem odděleného mezerou
# (počítáno od 0), který je výstupem příkazu exec rsh...
set hostname [lindex [exec rsh $ip "show run | include hostname"] 1]
set int [exec rsh $ip "show ip int brief"]
```

```
# určíme počet IP adres
# llength : vrací počet řetězců oddělených mezerou
set length [llength $int]

# v cyklu provádíme ping zjištěných IP adres
while {$counter<=$length} {
set tmp [lindex $int $counter]
# pomocí regularní výrazu ověříme zda proměnná tmp obsahuje IP adresu
# regulární výrazy odpovídají reg. výrazům z unixového shellu, ty
# složitější pak odpovídají reg. výrazům používaným ve Vim editoru,
# nebo v grep utilitě.
if {[regexp "(^\[0-9]+\.\[0-9]+\.\[0-9]+\.\[0-9]+)" $tmp ]} {
# puts : výpis na výstup
puts "\n"
puts "**********************************"
puts "* Ping $hostname [lindex $int [expr $counter - 1]]"
puts "**********************************"
# provedeme ping
exec "ping $tmp"
}
# zvyšujeme počítadlo
incr counter
}
} else {
puts "\n\n"
puts "IP address $ip is not reachable\n"
}
}
```

c) Spuštění skriptu a výsledek

Načteme proceduru ping_all z TFTP serveru, uloženou v souboru tclscript.tcl

```
Rack1R1(tcl)#source tftp://10.0.0.1/tclscript.tcl
```

Poté spustíme proceduru s parametrem IP adresy sousedního routeru

```
Rack1R1(tcl)#ping_all 150.1.2.2

**********************************
* Ping Rack1R2 Ethernet0/0
**********************************
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

**********************************
* Ping Rack1R2 Loopback0
**********************************
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.1.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Rack1R1(tcl)#
```

4.2 SNMP a MIB příklad

Popis:
Pomocí SNMP příkazů získáme informace z MIB databáze o průměrných přenosových rychlostech na jednotlivých rozhraních. Hodnoty typu 1.3.6.1.2.1.2.1.0 atd. odpovídají umístění (odkaz) požadovaných informací.

a) TCL Skript

```
# získáme požadované údaje
# 1.3.6.1.2.1.2.1.0 počet interfaců daného zařízení
set ifnumstr [ snmp_getone public 1.3.6.1.2.1.2.1.0]
regexp {val='([0-9]*)'} $ifnumstr {} ifnum
# 1.3.6.1.2.1.2.2.1.2 pojmenování portu
# 1.3.6.1.4.1.9.2.2.1.1.6  5-minutový průměr downloadu, bits/s
# 1.3.6.1.4.1.9.2.2.1.1.8  5-minutový průměr uploadu, bits/s
set vals [ snmp_getbulk public  0 $ifnum \
1.3.6.1.2.1.2.2.1.2  1.3.6.1.4.1.9.2.2.1.1.6
1.3.6.1.4.1.9.2.2.1.1.8]
# pokud nastane chyba skončíme
if { [regexp -nocase {error} $vals] == 1 } {
```

```
puts "SNMP ERROR: $vals"
exit 0
}
# upravíme získaná data do požadované podoby
# regsub : substituce hledaného řetězce za jiný
regsub -all {'} $vals {} val2
regsub -all {\{<obj oid=} $val2 {} val3
regsub -all { val} $val3 {} val4
regsub -all {/>\}} $val4 {} val5
# split : rozdělí proměnnou na seznam, oddělovačem jsou v tomto
# případě mezery
set vals [split $val5]

puts "\nifIndex\tifDescr\t\tifInBps\tifOutBps"
#v cyklu naplníme proměnné, které poté vypíšeme
for {set i 0} {$i < 2*$ifnum-1} {incr i 3} {
regexp {=([A-Za-z/0-9]*)} [lindex $vals $i] {} ifname
regexp {=([0-9]*)} [lindex $vals [expr 1+$i]] {} ifInBitsSec
regexp {=([0-9]*)} [lindex $vals [expr 1+$i]] {} ifInBitsSec
if {[string length $ifname] < 6} {
set ifname "$ifname\t"
}
# expr : vypočtení výrazu
puts "[expr $i/2]\t$ifname\t$ifInBitsSec\t$ifOutBitsSec"
}
```

b) Spuštění skriptu a výsledek

Skript je spuštěn z TFTP serveru a ze souboru pokus.tcl:

```
Router(tcl)#source tftp://10.0.0.1/pokus.tcl

Výpis:
Loading pokus.tcl from 10.0.0.1 (via Ethernet0/0):
[OK - 992 bytes]

ifIndex ifDescr         ifInBps ifOutBps
0       Ethernet0/0     1000    2000
1       Serial0/0       0       0
```

Poznámka:
Při psaní TCL skriptů je nutné mít na paměti, že jazyk TCL je značně striktní a k vytvoření složitějších příkazů je nutná velmi dobrá znalost TCL. Například tato hlavička procedury **proc ping_all {ip}** je v pořádku. Tato hlavička **proc ping_all{ip}** už ne (chybí mezera mezi názvem procedury a závorkou).

5. Závěr

Implementace jazyka TCL do Cisco prvků je bezpochyby velkým přínosem. Využití mezi administrátory jistě najde při opakovaných konfiguracích či testech, kde by stačilo pouze měnit parametry procedur. Zde by stálo za úvahu vytvoření aplikace, která by prostřednictvím GUI umožnila vytvářet skripty prakticky bez znalostí TCL, alespoň pro často opakující se sady příkazů a konfigurací. Další možné využití je spojení TCL a EEM. V současnosti je možné pomocí EEM reagovat na různé událostí spuštěním TCL skriptu. Například spustit vlastní TCL skript při přihlášení uživatele. Je nutné se dále zmínit o problému při běhu skriptů na některých routerech, které sice splňovaly svým IOS požadavky na bezchybný běh TCL skriptů, ale výsledkem byly chyby, týkající se proměnných (údajná neexistence proměnných). Při testu těchto skriptů na jiném routeru s novějším IOS bylo vše v pořádku. Administrátor používající TCL skripty by měl mít slušné povědomí o syntaxi tohoto jazyka. Jak již bylo zmíněno, chyby ve skriptech mohou vést až „k zamrznutí“ routeru a tedy nutnosti HW restartu prvku. Bohužel však zřejmě neexistuje vhodný simulační program, který by umožňoval danou syntaxi otestovat nanečisto. Jak vidno TLC pod Cisco prvky má své zápory, ale také nesporné klady. Záleží na schopnostech každého administrátora, zda je schopen najít a využít přednosti TCL ve spojení s Cisco prvky.

6. Literatura

[1] Cisco IOS Scripting with Tcl – referenční Cisco dokumentace http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a00801a75a7.html [28. 5. 2007]

[2] TCL'ing Your Cisco Router – vlastní popis podle Dr. Peter J. Welchera http://www.netcraftsmen.net/welcher/papers/iostcl01.html [28. 5. 2007]

[3] CCIE Lab Preparation Resources – příklad použití TCL skriptu http://www.internetworkexpert.com/resources/tclshrsh.htm [28. 5. 2007]

[4] Cisco IOS Software Embedded Event Manager, Harnesses Network Intelligence to Increase Availability – informace ohledně EEM http://www.cisco.com/en/US/products/ps6815/products_white_paper0900aecd803a4dad.shtml [28. 5. 2007]