Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Ústav aplikované informatiky

Využití skriptovacího jazyka TCL při přípravě síťových zařízení pro výuku v laboratorním prostředí

Bakalářská práce

Jan Tůma

Vedoucí práce: Ing. Rudolf Vohnout

České Budějovice 2014

Bibliografické údaje

Tůma, J., 2014: Využití skriptovacího jazyka TCL při přípravě síťových zařízení pro výuku v laboratorním prostředí. [The usage of TCL script language in network devices preparation for education in laboratory environment. Bc. Thesis, in Czech.] – 52 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

NÁZEV:

Využití skriptovacího jazyka TCL při přípravě síťových zařízení pro výuku v laboratorním prostředí.

ABSTRAKT:

Náplní této bakalářské práce je tvorba sady skriptů pro konfiguraci aktivních síťových prvků firmy Cisco, stručný popis skriptovacího jazyka TCL, využitých technologií a základních příkazů, které byly využity. Dále popis funkce jednotlivých skriptů a jejich možného využití v rámci výuky předmětů, využívajících tyto aktivní prvky.

KLÍČOVÁ SLOVA:

skript, jazyk TCL, Cisco IOS, Konfigurační průvodce, alias

TITLE:

The usage of TCL script language in network devices preparation for education in laboratory environment.

SUMMARY:

The contents of this bachelor thesis are to create a set of scripts for Cisco active network devices configuration, briefly describe TCL script language, used technologies and the main commands which were used. Further describe features of each script and theirs possible usage in subjects which use these active devices.

KEYWORDS:

script, TCL language, Cisco IOS, Configuration guide, alias

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 4. 4. 2014 Jan Tůma

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce panu Ing. Rudolfu Vohnoutovi za cenné rady a umožnění přístupu k aktivním prvkům. Dále bych rád poděkoval své rodině za podporu.

Obsah

1. Úvod

Snad každému správci sítě se nejspíše již stala situace, že potřeboval některé úkony na aktivních prvcích provádět opakovaně, ať již jde o jejich správu či získávání informací. Psaní stále stejných příkazů představuje samozřejmě nejen velkou ztrátu času, ale i narůstající nároky na pozornost, soustředění a z toho plynoucí vzrůstající možnost chyby při zadávání stále stejných příkazů. Proto již nejednoho jistě napadla otázka, jak si tuto práci ulehčit, nejlépe, jak ji zautomatizovat.

Ti správci sítě, kteří spravují síť s aktivními prvky od firmy Cisco, využívajících Cisco IOS[1] odpovídající verze, mají vynikající možnost napsat si na tyto činnosti vlastní skript a to pomocí skriptovacího jazyka TCL[2].

Pomocí tohoto skriptovacího jazyka lze tyto činnosti automatizovat. Např. skript na kompletní autokonfiguraci aktivního prvku. Správce jen spustí skript a prvek je nakonfigurován přesně jako ostatní prvky, které konfiguroval předtím. Pak ručně dokonfiguruje jen detaily, vztahující se pouze k danému aktivnímu prvku. Nehrozí zde žádná chyba při základní konfiguraci a její, někdy zdlouhavé, objevování a napravování. Dále je možné za pomoci skriptu reagovat na události.

Jde tedy o velmi mocný nástroj na zvýšení produktivity práce a správce se tak může věnovat důležitějším věcem. Např. ne sběru dat z prvků, ale jejich vyhodnocování a provádění opatření. Lze takto provádět takřka cokoli, co si správce dokáže naskriptovat.

Psaní si vlastních skriptů je samozřejmě o tolik výhodnější, o kolik je v síti více aktivních prvků, na kterých je třeba danou činnost provádět.

Pro zjednodušení se dále v této práci pro aktivní síťové prvky firmy Cisco s operačním systémem Cisco IOS používá pouze označení prvek. Tímto označením se tedy vždy myslí právě aktivní prvky Cisco s operačním systémem Cisco IOS.

V práci je nejprve nastíněn konkrétní řešený problém a jsou přiblíženy využité technologie. Pokračuje navržením vhodného řešení a samotnou implementací, ve které jsou zevrubně popsány funkce jednotlivých skriptů. Výstupem práce jsou, kromě samotných skriptů, manuál k jejich zprovoznění na prvku a kompletní dokumentace, která podrobně popisuje jejich konkrétní části.

[1] viz. Seznam použitých zkratek – IOS
[2] viz. Seznam použitých zkratek – TCL

1.1 Definice problému

Hlavním podnětem ke vzniku této bakalářské práce je nutnost opakované základní konfigurace prvku pro danou problematiku při výuce předmětů s těmito prvky. Tím se bohužel ztrácí čas při cvičeních, kdy příprava prvku na řešení zabere nejvíce času a následkem je bohužel nedostatek prostoru pro řešení skutečné náplně cvičení.

1.2 Cíle

Hlavním cílem této práce je tvorba sady skriptů v jazyku TCL pro použití na prvcích v rámci výuky předmětů využívajících tyto prvky. Tato sada skriptů dohromady slouží jako Konfigurační průvodce, nastavuje zálohování prvku a po ukončení práce ho připraví na další výuku.

Dále se tato práce zabývá samotným skriptovacím jazykem TCL, jeho historií, použitím v operačním systému Cisco IOS a popisem vybraných použitých příkazů tohoto jazyka. V dalších částech jsou popsány použité technologie, vytvořené skripty a jejich možné využití při výuce předmětů, které tyto prvky využívají.

2. Metodika

Při přípravě této bakalářské práce bylo nejprve nutné seznámit se důkladně s problematikou opakované konfigurace prvků při výuce a poté navrhnout vhodné skripty, které zjednoduší tuto rutinní práci s prvkem na možné minimum.

Dále bylo nutné seznámit se s operačním systémem Cisco IOS, a to hlavně z hlediska provádění konfiguračních příkazů a možností automatizování činností.

Při navrhování skriptů tedy byly hlavně zohledněny činnosti, které se provádějí opakovaně při většině bloků výuky, ale byly zahrnuty i činnosti ne tak časté, ale opakující se v rámci bloku a dále činnosti, které by mohly způsobovat nepříjemné chyby, které se hůře odhalují. Tyto chyby jsou přitom většinou způsobeny nepozorností a jejich odhalování tak zbytečně zdržuje. Za všechny by se dala zmínit alespoň např. nutnost zapnout konfigurovaný interface na směrovači, na rozdíl od přepínače. Pokud si toto uživatel neuvědomí, tak i když má vše nastaveno správně, směrovač nekomunikuje se sítí.

Psaní skriptů probíhalo v jazyce TCL, který je implementován od verzí operačního systému Cisco IOS 12.3(2)T, 12.2(25)S a 12.4 výše[3] a je tedy možné skripty napsané v tomto jazyce na daných prvcích spouštět.

[3]Securing Tool Command Language on Cisco IOS. *Security Intelligence Operations* [online]. © 1992–2013 [cit. 2014-03-10]. Dostupné z: `http://www.cisco.com/web/about/security/intelligence/securetcl.html`.

3. Přehled současného stavu

Prvky mají již v základu inicializačního konfiguračního průvodce. Tímto průvodcem se však dá nastavit jen velmi omezená množina nastavení a navíc je v anglickém jazyce, s čímž mohou mít někteří uživatelé problém.

Pro použití při výuce je tudíž tento průvodce nevhodný, protože neobsahuje konfiguraci částí, které jsou ve výuce opakovaně využívány. Tím zde vzniká prostor pro vznik vlastního průvodce, pomocí kterého se nastaví požadované základní a opakující se části konfigurace a další, již konkrétní, části týkající se aktuálního výkladu jsou konfigurovány pomocí příkazové řádky.

4. Přehled využitých technologií

Na začátku této kapitoly je uveden souhrnný seznam všech využitých programů a technologií, přičemž technologie jsou dále rozebrány, včetně případných alternativ.

Technologie:

- **Skriptovací jazyk TCL**
- **Plánovač příkazů Kron**
- **Kompilace skriptů do bajtkódu**
- **Aliasy**

Programy:

- **7-Zip 9.20** – Kompresní Open-Source program, který umožňuje kompresi souborů a jejich zabalení do archivu. Byl využit pro tvoření .tar archivů a je zdarma dostupný na `http://www.7-zip.org/`.
- **Komodo Edit 8.5.3** – Jedná se o Open-Source editor, který podporuje množství jazyků (mimo jiné právě TCL) a byl použit pro psaní skriptů. Je zdarma dostupný na `http://komodoide.com/download/#edit`.
- **TclPro 1.4.1** – Program umožňující převod skriptů z jazyka TCL do bajtkódu, který je funkční v operačním systému Cisco IOS[4]. Byl použit pro převod skriptů do bajtkódu. Původně se jednalo o komerční software[5], nyní je však ke stažení zdarma na `http://www.tcl.tk/software/tclpro/eval/1.4.html`.
- **TeraTerm Pro Web 3.1.3** – Terminál pro připojení k zařízení, ať již vzdáleně pomocí Telnetu/SSH nebo lokálně přes RS232. Byl využit pro práci s prvkem a testování skriptů na prvcích. Je zdarma dostupný na `https://www.ayera.com/teraterm/`.

[4]BLAIR, Ray, Arvind DURAI a John LAUTMANN. *Tcl scripting for Cisco IOS*. p. 282.
[5]Download TclPro 1.4. *Tcl Developer Xchange* [online]. 2001 [cit. 2014-03-10]. Dostupné z: `http://www.tcl.tk/software/tclpro/eval/1.4.html`.

- **Tftpd32 v3.0** – TFTP server, který byl využit pro vzdálený přístup ke skriptům z prvku a jejich kopírování na prvek. Ke stažení zdarma na `http://tftpd32.jounin.net/tftpd32_download.html`.

- **Robodoc 4.99.41** – nástroj na generování dokumentace ze zdrojových kódů. Umožňuje zápis dokumentačních komentářů přímo do zdrojového kódu a velkou upravitelnost podle vlastních potřeb. Podporuje velké množství programovacích jazyků. Byl využit na vygenerování dokumentace ze všech skriptů a je zdarma dostupný na `http://rfsber.home.xs4all.nl/Robo/index.html`.

4.1 Skriptovací jazyk TCL

Skriptovací jazyk TCL byl vytvořen panem Johnem Ousterhoutem v roce 1988 na základě práce, kterou vykonával na University of California v Berkeley, USA od roku 1980[6]. Jedná se o interpretovaný programovací jazyk, který, na rozdíl od kompilovaného, je překládán až těsně před spuštěním programu. Jazyk je beztypový a vše je považováno za řetězce[7].

Mezi hlavní výhody jazyka patří:

- jednoduchá tvorba aplikací[7]
- čitelnost kódu pro ostatní uživatele[8]
- nezávislost na platformě[7]
- rozšiřitelnost[6]
- rychlá naučitelnost[8]
- podpora kódování Unicode[7]

[6]OUSTERHOUT, John. History of Tcl. *Tcl Developer Xchange* [online]. 2001 [cit. 2013-06-05]. Dostupné z: `http://www.tcl.tk/about/history.html`.
[7]TIŠNOVSKÝ, Pavel. Programovací jazyk TCL. *ROOT.CZ* [online]. 2005 [cit. 2014-03-10]. Dostupné z: `http://www.root.cz/clanky/programovaci-jazyk-tcl/`.
[8]BLAIR, Ray, Arvind DURAI a John LAUTMANN. *Tcl scripting for Cisco IOS.* p. 1–2.

Naopak jako nevýhody lze označit:

- pomalejší běh programu z důvodu nutnosti jeho přeložení před samotným provedením[8]
- větší paměťové nároky z důvodu mít v operační paměti načten, kromě proměnných, i zkompilovaný kód[8]
- čitelnost kódu může být zároveň nevýhodou v komerční sféře, ve které je třeba chránit obsah před kopírováním[8]

Tento jazyk umožňuje i tvorbu grafického rozhraní, a to pomocí komponenty Tk[9], která byla dokončena v roce 1990[6].

Jazyk TCL je využíván v mnoha oblastech od webových aplikací, přes testování a automatizaci, vestavěné systémy, až po databáze[10]. Je udržován komunitou vývojářů a distribuován pod Open-Source licencí Tcl/Tk license, která umožňuje i bezplatné komerční využití[11]. Standardní příponou TCL skriptů je .tcl.

Pro tvorbu opakovatelně použitelných knihoven procedur poskytuje jazyk TCL tvorbu balíčků – packages. Tyto balíčky obsahují procedury, které mohou být po připojení balíčku volány[12]. Jedná se tak vlastně o obdobu dynamicky připojovaných knihoven (.dll v OS Windows), využívaných např. v jazyku C#.

4.2 Převod skriptů do bajtkódu

Vytvořené skripty v jazyku TCL, lze převést do bajtkódu. Sice není tímto získána zaznamenatelná výkonová výhoda[4], ale je tím alespoň zabráněno manipulaci s kódem skriptu, čímž by mohlo dojít k problémům při vykonávání kódu skriptu. I proto jsou výsledné skripty, které by měly být umístěny na prvku, převedeny do bajtkódu. Převedený skript má příponu .tbc.

[9] viz. Seznam použitých zkratek – Tk

[10] Uses for Tcl/Tk. *Tcl Developer Xchange* [online]. 2001 [cit. 2013-06-05]. Dostupné z: `http://www.tcl.tk/about/uses.html`.

[11] Support and More. *Tcl Developer Xchange* [online]. 2001 [cit. 2013-06-05]. Dostupné z: `http://www.tcl.tk/about/support.html`.

[12] Building reusable libraries – packages and namespaces. FLYNT, Clif. *Tcl Tutorial* [online]. [2003] [cit. 2014-03-10]. Dostupné z: `http://www.tcl.tk/man/tcl8.5/tutorial/Tcl31.html`.

Důležitou věcí před převodem do bajtkódu je, pokud je někde ve skriptech odkaz na jiné skripty, které budou také převedeny, změnit všechny přípony z .tcl na .tbc. V opačném případě nebudou odkazy funkční.

Přílohou této bakalářské práce jsou tak dvoje skripty. Ve složce skripty/nezkompilované jsou umístěny nezkompilované verze skriptů (přípona .tcl) a ve složce skripty/zkompilované jsou zkompilované skripty (přípona .tbc).

4.3 TCL interpret v Cisco IOS

Implementace TCL interpretu do operačního systému Cisco IOS přinesla zavedení několika nových rozpoznávaných příkazů. Pro účely této práce se jedná zejména o příkaz „ios_config“[13].

Na prvku je interpret spouštěn pomocí příkazu „tclsh“ v privilegovaném režimu[13]. Po spuštění interpretu tímto příkazem jsou všechny zavedené proměnné či procedury drženy v paměti, a to až do opuštění interpretu. Toto se týká i skriptů, tudíž do konce provedení skriptu je všechno provedené tímto skriptem drženo v paměti a pokud se ve skriptu spouští další skripty, jedná se stále o jeden interpret, což je nutné mít na paměti.

Užitečným příkazem Cisco IOS je „scripting tcl init skript“, který umožňuje spouštění určitého skriptu vždy, pokud je spouštěn TCL interpret[13]. Toho lze využít pro načtení balíčku, když Cisco IOS nepodporuje vyhledávání balíčků v předem definovaných umístěních[14].

Rozpoznávání příkazů zadaných do TCL interpretu probíhá tak, že se nejprve zjišťuje, zda jde o TCL příkaz a pokud ne, zda jde o Cisco IOS příkaz. Takto tedy příkazy propadávají, nicméně je vhodnější, pokud se zadává IOS příkaz v TCL interpretu, použít příkaz „exec“. Nedochází tak ke zbytečnému zatěžování prvku[15].

[13] Cisco IOS Scripting with Tcl. *Cisco IOS Scripting with TCL Configuration Guide, Cisco IOS Release 12.4T* [online]. 2011 [cit. 2014-03-10]. Dostupné z: `http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ios_tcl/configuration/12-4t/ios-tcl-12-4t-book/nm-script-tcl.html`.

[14] PEPELNJAK, Ivan. Using Tcl packages on Cisco IOS. *IpSpace* [online]. 2007 [cit. 2014-03-10]. Dostupné z: `http://blog.ipspace.net/2007/09/using-tcl-packages-on-cisco-ios.html`.

[15] BLAIR, Ray, Arvind DURAI a John LAUTMANN. *Tcl scripting for Cisco IOS.* p. 34–36.

4.4 Možnosti automatického spouštění skriptů

Plně automatického spouštění skriptu nebo zadané sekvence příkazů lze docílit v zásadě dvěma způsoby, a to pomocí plánovače úloh Kron nebo EEM[16].

Kron je méně propracovaný a v nižších verzích Cisco IOS umožňuje pouze spuštění určité události v zadaný čas, a to jednou nebo opakovaně. V novějších verzích umožňuje i spuštění události po startu prvku[17]. Výhodou Kronu je, že se většinou nachází i ve verzích Cisco IOS, ve kterých není dostupný EEM[18].

EEM je naproti tomu velmi propracovaný a kromě funkcí společných s Kronem umožňuje i reakce na události[19]. Jeho již zmíněnou nevýhodou je, že je přítomen na menším počtu prvků, než Kron. Strukturu EEM znázorňuje obrázek 4.1, na kterém jsou vidět 3 základní bloky – politiky, EEM server a detektory událostí.

4.5 Automatické zálohování konfigurace

Při konfiguraci prvku nejsou provedené změny trvalé. Aby byly změny trvalé, je třeba uložit aktuální konfiguraci, a to do startovní konfigurace nebo do souboru v paměti prvku, odkud může být načtena. Pokud tedy během výuky uživatel omylem restartuje prvek nebo vypadne proud, je dosavadní konfigurace ztracena. To může být velmi nepříjemné, zvláště pokud již bylo provedeno množství konfiguračních příkazů.

Skripty Konfiguračního průvodce tedy nastavují ukládání aktuální konfigurace do záložního souboru, aby mohla být konfigurace případně obnovena. Docílit automatického zálohování aktuální konfigurace lze pomocí Kronu i EEM.

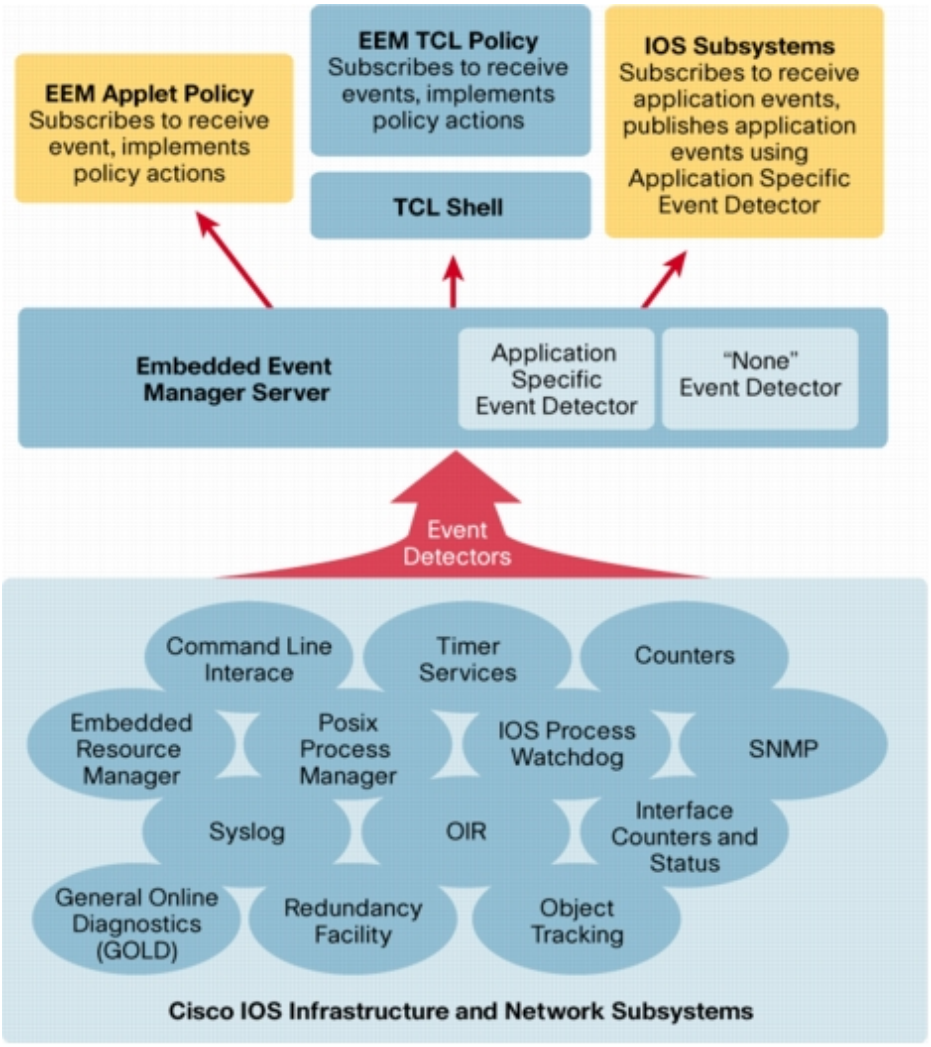
[16] viz. Seznam použitých zkratek – EEM

[17] Command Scheduler (Kron). *Cisco Networking Services Configuration Guide, Cisco IOS XE Release 3S (Cisco ASR 1000)* [online]. © 1992–2013 [cit. 2014-03-13]. Dostupné z: `http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cns/configuration/xe-3s/asr1000/cns-xe-3s-asr1000-book/cns-cmd-sched.html`.

[18] PEPELNJAK, Ivan. Kron: poor-man's cron. *IpSpace* [online]. 2007 [cit. 2014-03-13]. Dostupné z: `http://blog.ipspace.net/2007/11/kron-poor-man-cron.html`.

[19] Cisco IOS Embedded Event Manager (EEM). *Management Instrumentation* [online]. © 1992–2013 [cit. 2014-03-13]. Dostupné z: `http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-embedded-event-manager-eem/index.html`.

[20] EEM System Architecture [obrázek]. *Cisco IOS Software Release 12.2SB New Features and Hardware Support, Product Bulletin 3258* [online]. 2008 [cit. 2014-03-10]. Dostupné z: `http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-software-releases-12-2-s/prod_bulletin0900aecd80586fe2.html`.



Obrázek 4.1: Blokové schéma EEM[20]

4.6 Aliasy v Cisco IOS

V operačním systému Cisco IOS mohou být definovány vlastní příkazy, ale je možné předefinovat i již integrované příkazy. Toho lze využít k tomu, aby po zadání určitého příkazu došlo k vykonání požadované akce[21]. Lze tak např. zajistit, aby po zadání příkazu „pruvodce“ došlo ke spuštění skriptů Konfiguračního průvodce. V tomto případě se jedná o definování nového rozpoznávaného příkazu.

[21] A through B: alias. *Cisco IOS Configuration Fundamentals Command Reference* [online]. © 1992–2013 [cit. 2014-03-19]. Dostupné z: `http://www.cisco.com/c/en/us/td/docs/ios/fundamentals/command/reference/cf_book/cf_a1.html#wp1013037`.

Pokud je předefinován stávající příkaz, je možné zajistit, aby uživatel používal již naučený příkaz, ale prováděla se jiná akce.

Při definování aliasů je třeba vzít v úvahu, že vytvořený alias platí vždy na přesně daný řetězec, a proto, pokud byl např. definován alias „pruvodce“, příkaz „pruvodc“ nebude rozpoznán, protože neodpovídá přesně šabloně. Toto je důležité především pokud je předefinováván již integrovaný příkaz, kterému má být touto cestou změněna funkčnost.

5. Návrh řešení

Skripty byly navrženy způsobem, že nejprve jsou uživateli pokládány otázky a sbírány jeho odpovědi. Tyto odpovědi jsou ukládány do listů a až na konci skriptu je provedena samotná konfigurace prvku. V úvahu připadala ještě varianta, ve které by rovnou po sesbírání určitého počtu odpovědí došlo ke konfiguraci. Výhodou druhého řešení by byla možnost okamžité reakce na případné chyby, nicméně tento způsob nebyl zvolen z důvodu tříštění pokládaných otázek. Tím by se narušila jednoduchost průvodce a navíc při výuce tato základní konfigurace většinou probíhá podle zadání, tudíž je riziko chyby nižší. Další nevýhodou tohoto řešení by bylo, že pokud by skript skončil dříve, ať už kvůli jeho pádu nebo ukončení uživatelem, došlo by k situaci, kdy prvek by již byl z části nakonfigurován, ale nebylo by příliš zřejmé, kde konfigurace skončila. Naopak při řešení s konfigurací prvku až na konci skriptu je toto eliminováno, protože až do jeho konce není provedena žádná změna v konfiguraci.

Vzhledem k tomu, že skripty mají většinou podobný charakter, tj. položení otázky, sběr odpovědi a nakonec konfigurace, byl z důvodu minimalizace opakovaných částí kódu zvolen přístup, ve kterém jsou tyto opakující se procedury uloženy v samostatném skriptu a tento funguje jako balíček, který je poskytuje ostatním skriptům.

Pro zjednodušení konfigurace byla do skriptů, se kterými uživatel pracuje, implementována nápověda. Dále, aby nedocházelo ke zbytečným chybám, je při sbírání odpovědi tento vstup validován a uživateli je dána možnost ho opravit. Případné chyby se tak omezují jen na specifické, např. přiřazení stejné IP adresy více interfacům. Tyto chyby jsou po dokončení konfigurace vypsány, aby o nich měl uživatel ponětí a mohl je opravit.

Při provádění skriptů je třeba rozlišit o jaký prvek se jedná. Je využito chybové hlášky při nerozpoznání požadovaného příkazu a jako vhodný příkaz byl vybrán „show vlan“, který nefunguje, pokud je prvkem směrovač. Podle toho je rozhodnuto, které části kódu budou vykonány.

Další částí, kterou bylo nutné vyřešit, bylo automatické zálohování aktuální konfigurace. Provést tuto činnost umožňují oba systémy – Kron i EEM. Bohužel EEM není součástí přepínačů 2960, které jsou ve výukových prostorách. Tudíž byl pro tuto činnost zvolen Kron, který je na všech současných výukových prvcích umístěn a pro tuto činnost plně dostačuje.

Posledním úskalím při návrhu skriptů byla otázka, jak docílit automatického spuštění Konfiguračního průvodce po startu prvku a zajistit nahrání předpřipravené konfigurace při příštím startu. Pro tento účel by byl vhodný EEM, který však bohužel na přepínačích není, jak již bylo poznamenáno výše. Nabízí se tedy řešení pomocí aliasů. Nešlo by tedy o automatické spouštění, ale o navázání na určitý konfigurační příkaz, který je vždy zadán. Byly vybrány příkazy „configure terminal“, který je vždy zadán, a příkaz „reload“. Jak bylo popsáno výše, při použití aliasů je nutné počítat se všemi možnostmi zapsání tohoto příkazu a je tedy nutné konfigurovat více aliasů.

Určitým specifikem je, že alias funguje jen pro první část příkazu (configure) a je tedy nutné druhou část (terminal), kterou je uživatel zvyklý zadávat, odfiltrovat pomocí „| exclude“. Pravděpodobně jedinou nevýhodou tohoto řešení je nutnost zadat příkaz „configure terminal“ dvakrát, protože při prvním zadání se spustí průvodce, ale uživatel se nedostane do konfiguračního režimu a musí tedy po skončení průvodce zadat příkaz znovu. Nicméně toto není pravděpodobně až takový problém, vzhledem k tomu, že k tomuto jevu dochází jen při prvním zadání tohoto příkazu po startu prvku, což se zpravidla děje jednou za celou dobu cvičení.

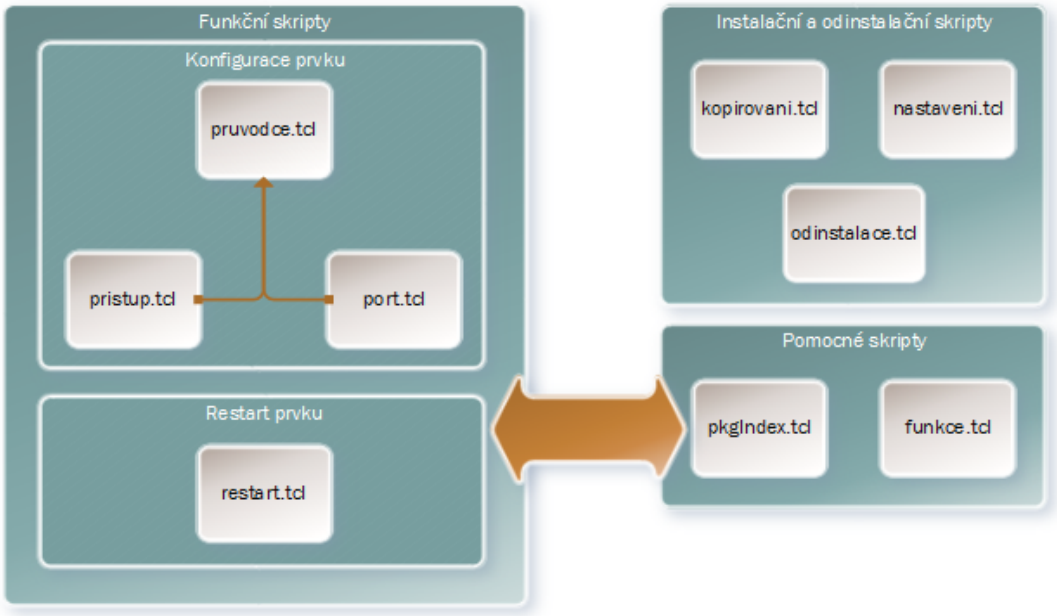
Jak již bylo naznačeno výše, aliasy je nutné opět rušit, aby byla uživateli umožněna normální činnost. K tomuto dochází vždy při spuštění daného skriptu. U skriptu pruvodce.tcl při jeho spuštění a u skriptu restart.tcl až za předpokladu, že chce uživatel prvek opravdu restartovat, aby nedocházelo k zacyklení. Aby měl uživatel později snadný další přístup k průvodci, je pro něj definován alias „pruvodce“.

Vzhledem k navázání skriptů na aliasy je vhodné při mazání skriptů z prvku postupovat doporučeným způsobem, který je uveden v manuálu Konfiguračního průvodce – viz. Přílohy – Konfigurační průvodce manuál. Pokud by byly skripty

jen smazány, může dojít k situaci, ve které nebude možný vstup do konfiguračního režimu a restart prvku, a to z důvodu absence požadovaného skriptu, který je pomocí aliasu na tento příkaz navázán.

5.1 Blokové schéma

Celkový koncept skriptů konfiguračního průvodce naznačuje blokové schéma na obrázku 5.1.



Obrázek 5.1: Blokové schéma skriptů Konfiguračního průvodce

Blok funkční skripty znázorňuje všechny skripty, se kterými přijde uživatel přímo do styku a implementují nápovědu i kontrolu neplatných vstupů. Tento blok je dále rozdělen na bloky Konfigurace prvku a Restart prvku. Blok Konfigurace prvku sdružuje skripty, které jsou využívány při konfiguraci Konfiguračním průvodcem. Tyto skripty jsou volány při prvním zadání příkazu „configure terminal“ a příkazem „pruvodce“. Blok Restart prvku obsahuje skript využitý pro restartování prvku. Ten je volán pouze příkazem „reload“. Celý blok Funkční skripty využívá procedur dostupných pomocí bloku Pomocné skripty. Blok Pomocné skripty pouze poskytuje pomocné procedury.

Posledním blokem je blok Instalační a odinstalační skripty. S těmi pracuje pouze správce prvků, který zavádí skripty na prvky. Slouží k usnadnění zavedení skriptů, připravení prvku do požadovaného stavu a případné odinstalaci. Pro odinstalaci byl definován příkaz „odinstalace“, po jehož zadání dojde ke spuštění příslušného skriptu a uvedení prvku do výchozího továrního stavu.

6. Implementace

V této kapitole jsou vysvětleny funkce jednotlivých skriptů a uvedeny základní použité příkazy. Je zde tedy u každého skriptu stručně uveden jeho účel a blokové schéma, zatímco kompletní popis jeho činnosti je součástí dokumentace. Kompletní dokumentace zdrojových kódů skriptů je součástí příloh, viz. Přílohy – Dokumentace. Aby skripty fungovaly společně jako Konfigurační průvodce, je třeba postupovat při jejich zavádění na prvek dle manuálu, který detailně popisuje kroky zprovoznění skriptů, jejich využití i odinstalaci. Tento manuál je také v přílohách, viz. Přílohy – Konfigurační průvodce manuál.

6.1 Zásadní použité příkazy

V této části jsou popsány zásadní příkazy, které byly použity při psaní skriptů. Nejedná se tedy o kompletní souhrn a nejsou zde uvedeny např. příkazy pro tvorbu podmínek, cyklů a další.

- **puts** – základní příkaz, umožňující výpis na konzoli. Má volitelný parametr -nonewline, který neprovede řádkový zlom.

- **gets stdin** – základní příkaz, který čte po řádku vstup určeného kanálu.

- **set** – základní příkaz pro deklaraci proměnné i změnu její hodnoty. Pokud jsou za název proměnné přidány {}, jedná se o deklaraci listu. List byl ve skriptech často využíván, a to i jako vnořený list v listu. Byl zvolen namísto pole, protože se jedná, na rozdíl od pole v TCL, o uspořádanou kolekci. Je k ní přistupováno pomocí číselného indexu a má menší paměťovou náročnost[22].

[22] Array. *The Tcler's Wiki* [online]. 2013 [cit. 2014-03-13]. Dostupné z: `http://wiki.tcl.tk/1032`.

Pro práci s listem je k dispozici množství funkcí, z nichž byly využity především:

  - *lappend* – přidání prvku na konec listu
  - *lindex* – přístup k prvku listu pomocí indexu
  - *llength* – získání počtu prvků v listu
  - *lreplace* – nahrazení určeného počtu prvků listu

- **ios_config** – slouží k provádění konfiguračních příkazů na prvku. Zadávají se pomocí něj příkazy globálního konfiguračního režimu. Pokud je třeba provést více příkazů najednou (případně se více zanořit, např. při konfiguraci interfaců), je třeba oddělit tyto příkazy od sebe pomocí uvozovek. Musí však být zadány najednou pomocí jednoho příkazu ios_config. Např. ios_config "interface f0/1" "shutdown" "end". Příkaz by měl vždy končit „end“[23].

- **exec** – slouží k provádění příkazů privilegovaného režimu a nelze pomocí něj konfigurovat prvek. Např. exec "show vlan". U standardního TCL interpretu (ne pod OS Cisco IOS) slouží tento příkaz ke spouštění programů.

6.2 Skript funkce.tcl

Skript sám neprovádí žádnou interakci s uživatelem, ale slouží jako knihovna procedur. Tyto procedury jsou využívány větším počtem skriptů, a proto jsou uloženy ve speciálním skriptu, aby nedocházelo k jejich zbytečné duplikaci. Skript, který využívá procedury z tohoto skriptu je importuje do svého jmenného prostoru příkazem „namespace import“ a na začátku je uvedeno, že ke svému spuštění skript vyžaduje daný balíček příkazem „package require“.

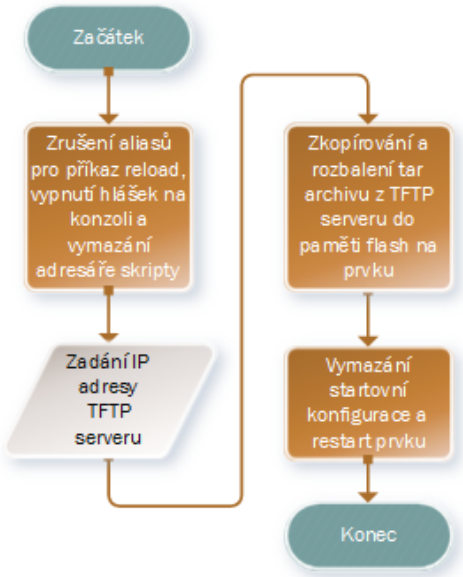
Obrázek 6.1 ukazuje, jak je tento skript využit dalšími skripty. Po zavolání požadované procedury se provede procedura z tohoto skriptu, skript skončí a vykonávání kódu pokračuje v původním skriptu, který zavolal proceduru.

[23]BLAIR, Ray, Arvind DURAI a John LAUTMANN. *Tcl scripting for Cisco IOS.* p. 48.

Obrázek 6.1: Blokové schéma skriptu funkce.tcl

6.3 Skript kopirovani.tcl

Skript není součástí skriptů kopírovaných na prvek a je využit jen pro zkopírování skriptů Konfiguračního průvodce na prvek. Kopírované skripty je třeba mít zabalené v .tar archivu. Jeho činnost znázorňuje obrázek 6.2.



Obrázek 6.2: Blokové schéma skriptu kopirovani.tcl

Po spuštění je nejprve třeba zadat adresu TFTP serveru, ze kterého budou ostatní skripty kopírovány na prvek. Po zkopírování jsou skripty rozbaleny do složky skripty do lokální paměti prvku. Nakonec skript vymaže konfiguraci prvku a restartuje ho. Vymazání konfigurace je důležité, protože v dalším kroku je nutné nastavit spouštění Konfiguračního průvodce a zálohování prvku skriptem nastaveni.tcl.

6.4 Skript nastaveni.tcl

Schéma tohoto skriptu je na obrázku 6.3.

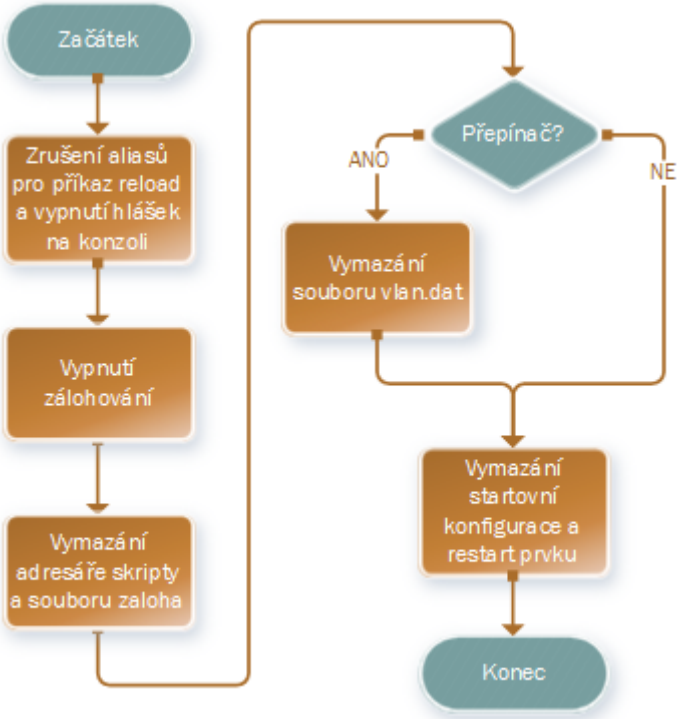
Obrázek 6.3: Blokové schéma skriptu nastaveni.tcl

Skript po spuštění provede všechny potřebné činnosti tak, aby byl zprovozněn Konfigurační průvodce, včetně zálohování. Nastavení pomocí tohoto skriptu je vhodné provádět na čisté konfiguraci, protože po nastavení prvku dojde k uložení

aktuální konfigurace a tato je později využívána jako startovní. Tudíž vše, co je aktuálně nastaveno na prvku, bude později nastaveno po každém restartu, a proto by neměly být nastaveny např. IP adresy interfaců.

6.5 Skript odinstalace.tcl

Skript je využit pro jednoduché odinstalování skriptů konfiguračního průvodce z prvku. Jeho schéma znázorňuje obrázek 6.4.



Obrázek 6.4: Blokové schéma skriptu odinstalace.tcl

Po spuštění skript provede všechny nezbytné kroky potřebné k úplné odinstalaci skriptů z prvku, včetně samotného vymazání skriptů, a uvede prvek do výchozího továrního nastavení.

6.6 Skript pkgIndex.tcl

Skript po spuštění zařadí balíček procedur funkce.tcl do balíčků, které mohou být později volány. Musí být spouštěn při každém spuštění TCL interpretu na prvku, aby mohl být balíček využit. Obrázek 6.5 znázorňuje činnost tohoto skriptu.

Obrázek 6.5: Blokové schéma skriptu pkgIndex.tcl

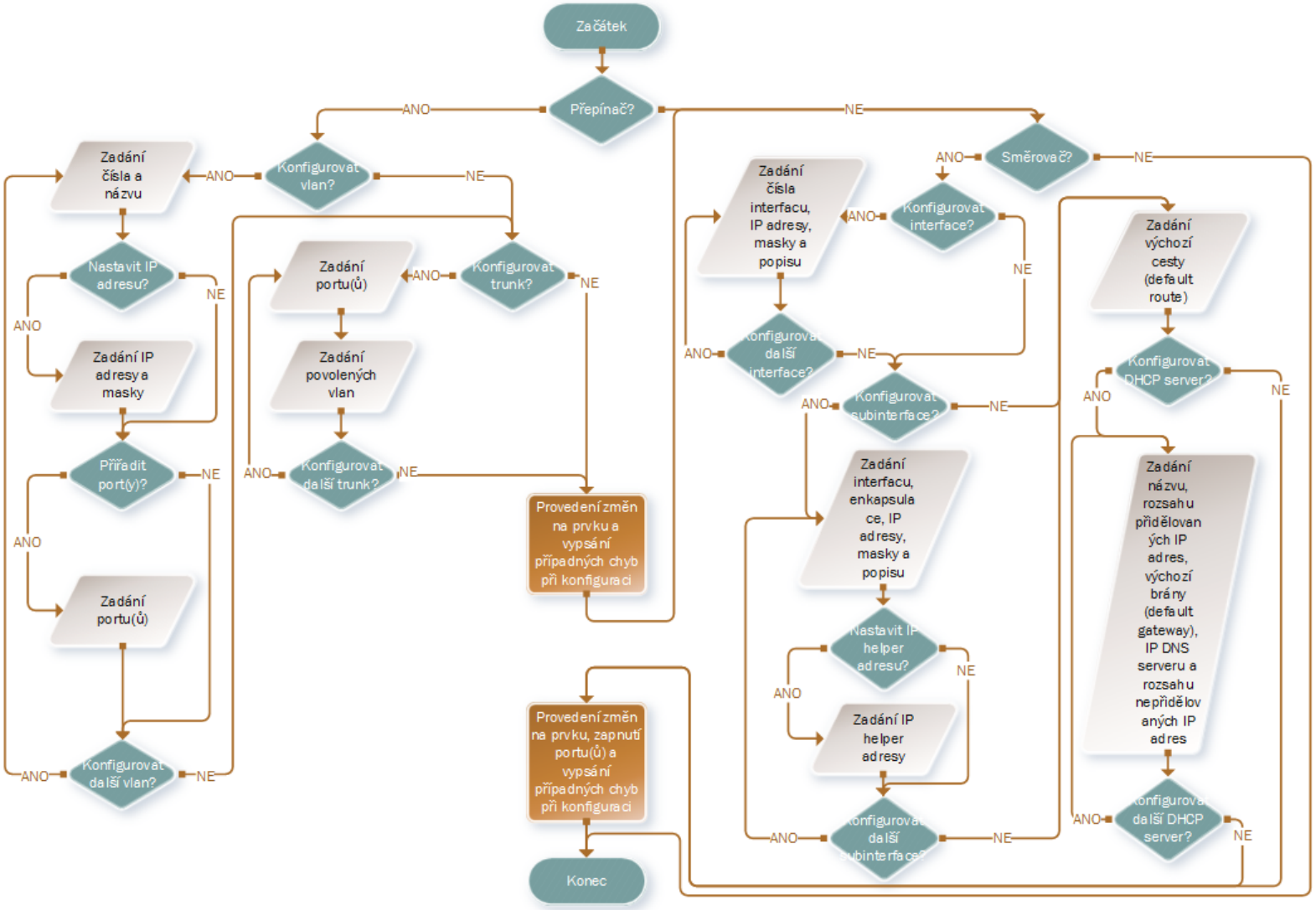
6.7 Skript port.tcl

Skript slouží ke konfiguraci interfaců, subinterfaců, vlan, trunků, výchozí cesty a DHCP serverů. Jde o důležitý skript, pomocí kterého je prováděna největší část konfigurace. Jeho celková činnost je vidět na obrázku 6.6.

Zajišťuje konfiguraci směrovače, přepínače i MLS[24], a to pomocí testování, o jaký se jedná prvek. Jako první probíhá dotazování se na požadované konfigurační změny a až po sesbírání informací dochází k samotné konfiguraci prvku.

Důležité je, že tento skript nemůže být samostatně spuštěn, protože sám neimportuje balíček procedur jantusCisco, který je k jeho běhu vyžadován.

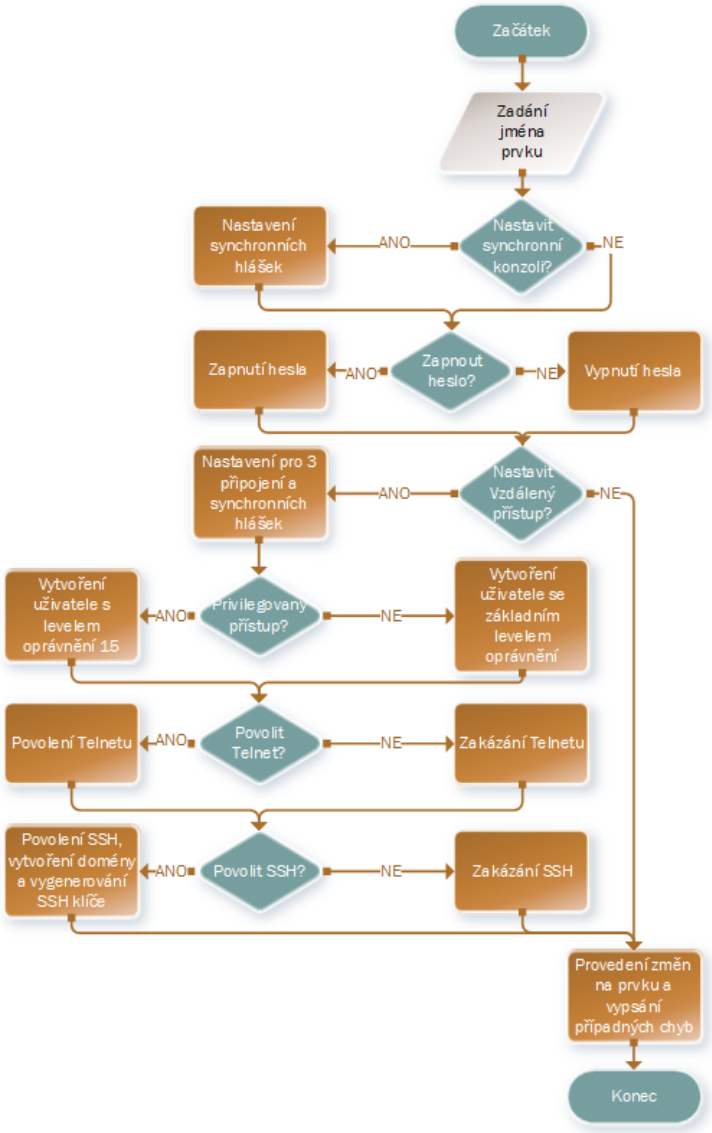
[24] viz. Seznam použitých zkratek – MLS



Obrázek 6.6: Blokové schéma skriptu port.tcl

6.8 Skript pristup.tcl

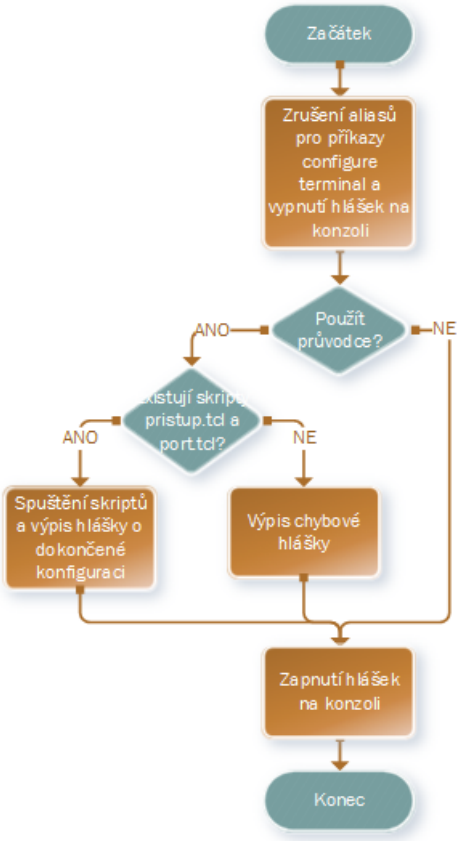
Skript umožňuje konfiguraci, týkající se lokálního i vzdáleného přístupu a základních věcí. Jeho činnost znázorňuje blokové schéma na obrázku 6.7 a stejně jako předchozí skript (port.tcl) nemůže být spouštěn samostatně, kvůli absenci importu balíčku jantusCisco.



Obrázek 6.7: Blokové schéma skriptu pristup.tcl

6.9 Skript pruvodce.tcl

Pomocí tohoto skriptu dochází ke spuštění Konfiguračního průvodce. Skript sdružuje skripty port.tcl a pristup.tcl, které jsou pomocí něj volány a vykonává jen doplňující funkce. Schéma tohoto skriptu ukazuje obrázek 6.8.



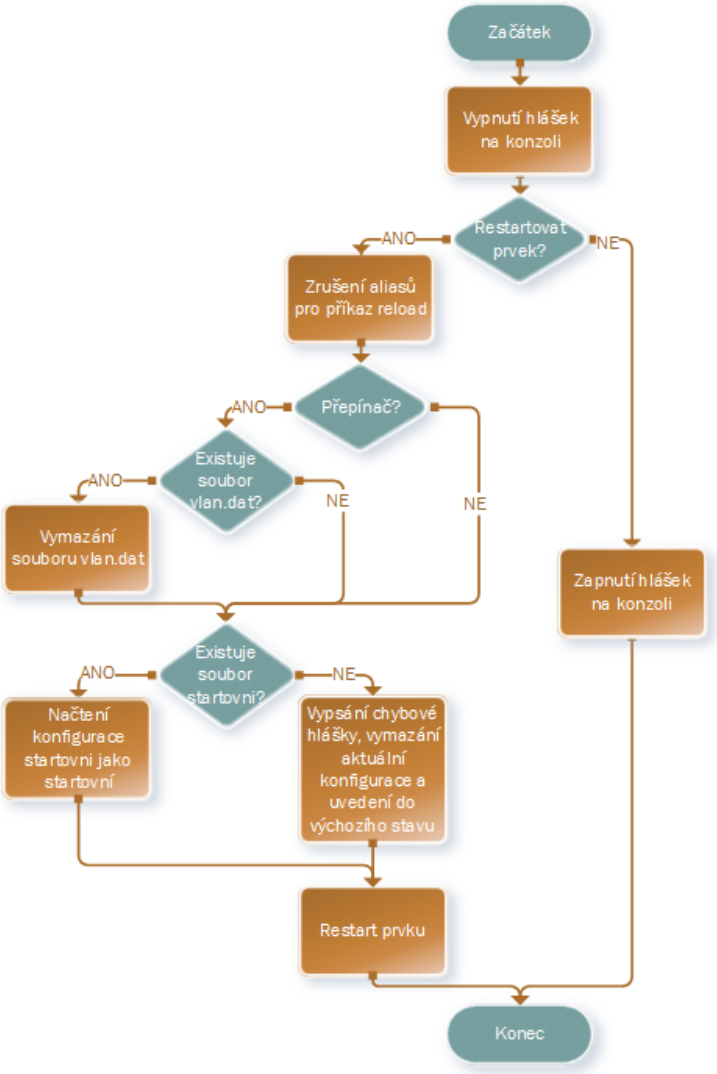
Obrázek 6.8: Blokové schéma skriptu pruvodce.tcl

Užitečnou funkcí tohoto skriptu je vypínání hlášek na terminálu. Nedochází tak k odskakování kurzoru a kouskování odpovědi. V opačném případě totiž dochází k výpisu hlášek i při běhu skriptu a jeho výstup by byl velmi nepřehledný.

Zároveň tento skript importuje balíček procedur jantusCisco, který využívají i volané skripty. Jimi již není importován, protože by došlo k chybě snahou importovat již existující procedury.

6.10 Skript restart.tcl

Tento skript není spouštěn společně s Konfiguračním průvodcem, ale je využíván až v případě, že uživatel chce restartovat prvek. Funkci skriptu ukazuje blokové schéma na obrázku 6.9.



Obrázek 6.9: Blokové schéma skriptu restart.tcl

Pokud má dojít k restartu prvku, skript vyčistí prvek od konfiguračních změn, včetně případných vlan a zavede jako startovní konfiguraci předpřipravenou. Když startovní konfigurace není, je prvek uveden do výchozího továrního nastavení.

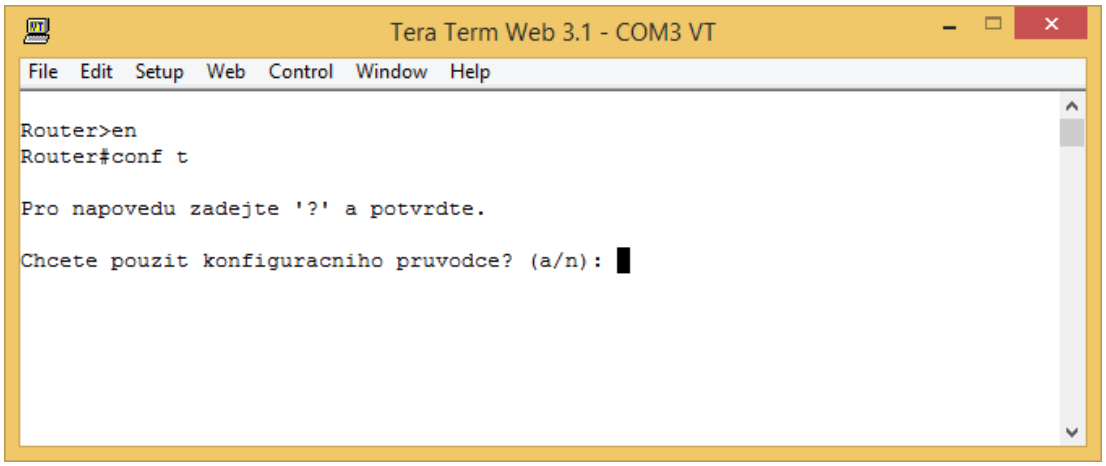
7. Možné využití skriptů

Skripty je možné využít při výuce všech předmětů, které využívají podporované prvky. Použitím těchto skriptů by mělo dojít k výraznému zkrácení doby nutné pro předkonfiguraci prvků a tím k nárůstu času věnovanému hlavní náplni cvičení. Tato část není členěna podle probíraných kapitol, jak je uvedeno ve vedlejších cílech, a to především z důvodu určité univerzálnosti vytvořeného průvodce, kdy jeho činnost není úzce spjata s určitým předmětem, ale umožňuje širší konfiguraci, a to především ve vztahu ke zdatnosti uživatele. Ten se tak může sám rozhodnout, které části chce raději konfigurovat sám a naopak, kdy je pro něj výhodnější použít průvodce. Této jeho univerzálnosti bylo dosaženo zařazením poměrně velkého množství základních konfiguračních možností.

7.1 Ukázka použití

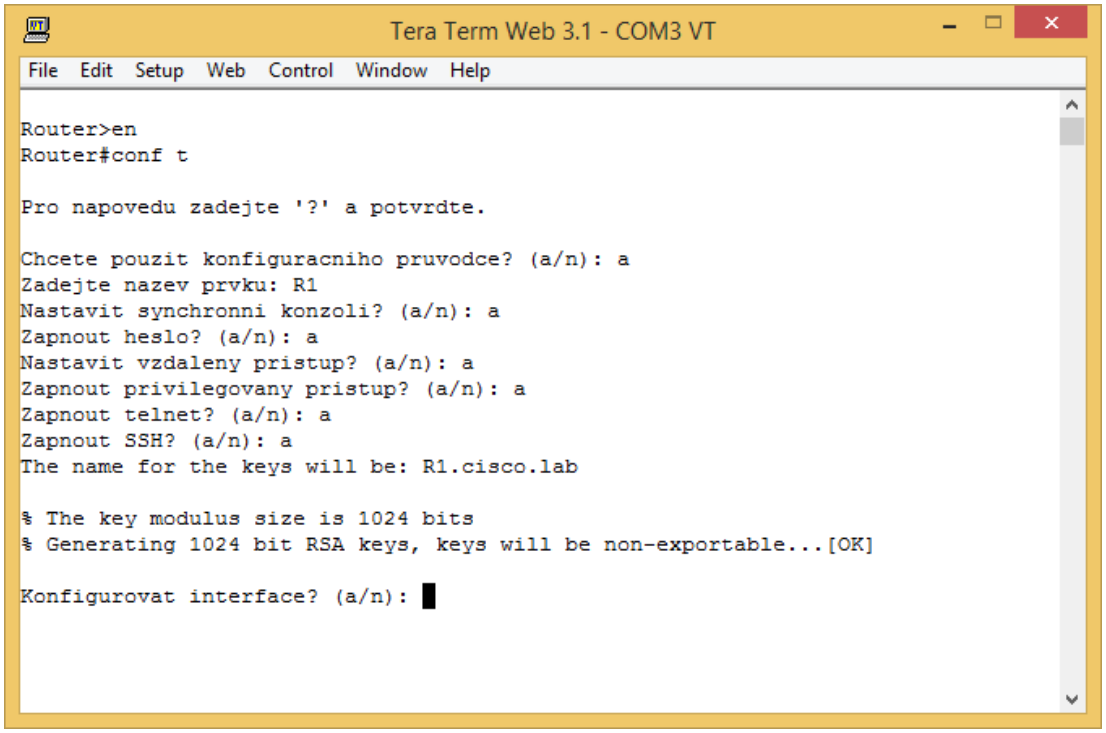
Následující obrázky ukazují, jak vypadá použití skriptů konfiguračního průvodce na prvku – směrovači.

Obrázek 7.1 zobrazuje prvek po startu. Uživatel zadal příkazy „enable“ a „configure terminal“. Tím došlo ke spuštění Konfiguračního průvodce a nyní si zvolí, zda ho použít, nebo ne. Pro nápovědu může zadat znak ’?’.



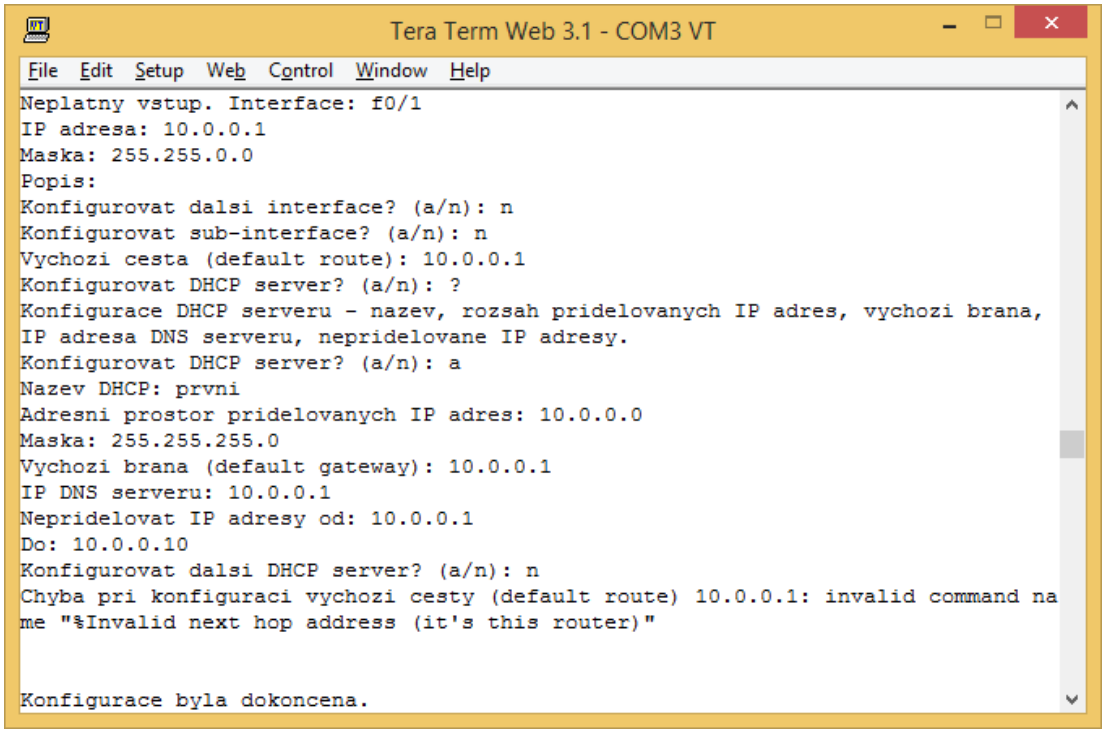
Obrázek 7.1: Ukázka konfigurace č. 1

Pokud uživatel zadá „ano“, průvodce pokračuje kladením otázek, které se vztahují k zabezpečení prvku a vzdálenému přístupu. Dále dochází ke konfiguraci interfaců prvku. Toto znázorňuje obrázek 7.2.



Obrázek 7.2: Ukázka konfigurace č. 2

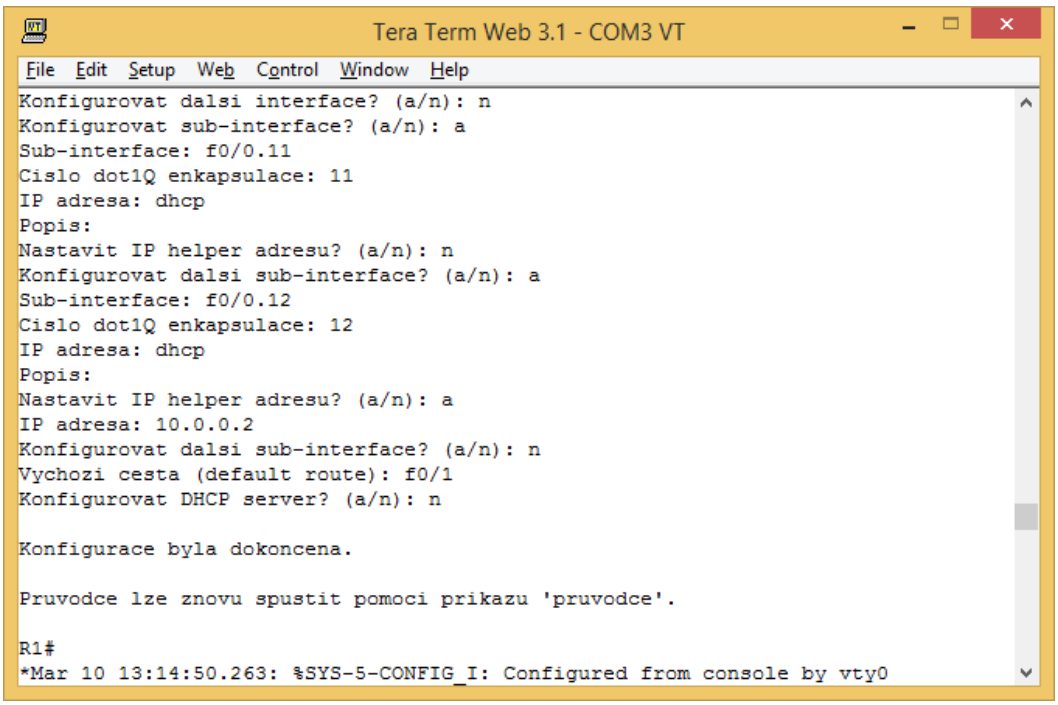
Obrázek 7.3 ukazuje konfiguraci interfacu. Nejprve byl zadán neplatný vstup. Po opravení vstupu a zadání vlastností interfacu pokračuje konfigurace dalšími interfacy a subinterfacy. Dále je nastavena výchozí cesta. Jako další krok může být nastaven DHCP server a je zobrazena nápověda, co vše se případně bude nastavovat. Uživatel pokračuje konfigurací DHCP serveru. Nakonec při provádění konfigurace je zobrazena chybová hláška. Došlo k ní, protože IP adresa výchozí cesty byla stejná, jako byla přidělena interfacu f0/1.



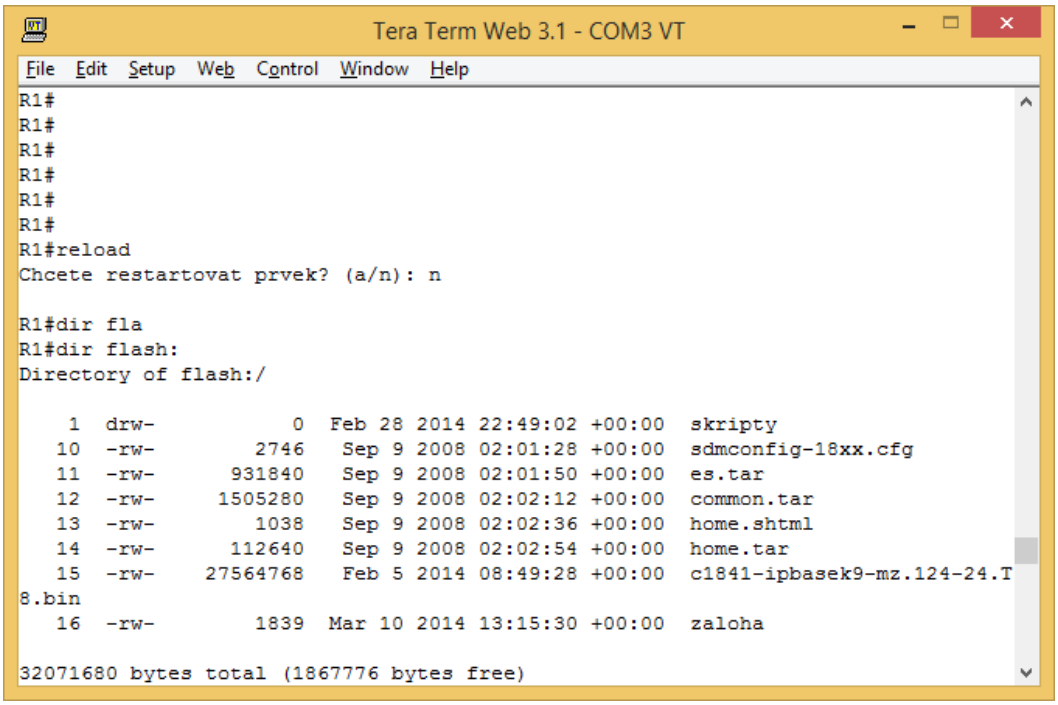
Obrázek 7.3: Ukázka konfigurace č. 3

Obrázek 7.4 je alternativou k předchozímu obrázku 7.3. Tentokrát byly nastavovány i subinterfacy a naopak nebyl konfigurován DHCP server. Při zadávání výchozí cesty byla již zadána správně, tudíž při provádění konfigurace nedošlo k chybě a byla již jen vypsána hláška o dokončené konfiguraci.

Obrázek 7.5 již nemá spojitost s předchozími, ale ukazuje funkci skriptů při restartování prvku. Po zadání příkazu „reload“, je uživatel dotázán, zda si opravdu přeje restartovat prvek. Dále je vidět soubor „zaloha“, do kterého je každých 5 minut zálohována aktuální konfigurace a může být odtud obnovena.

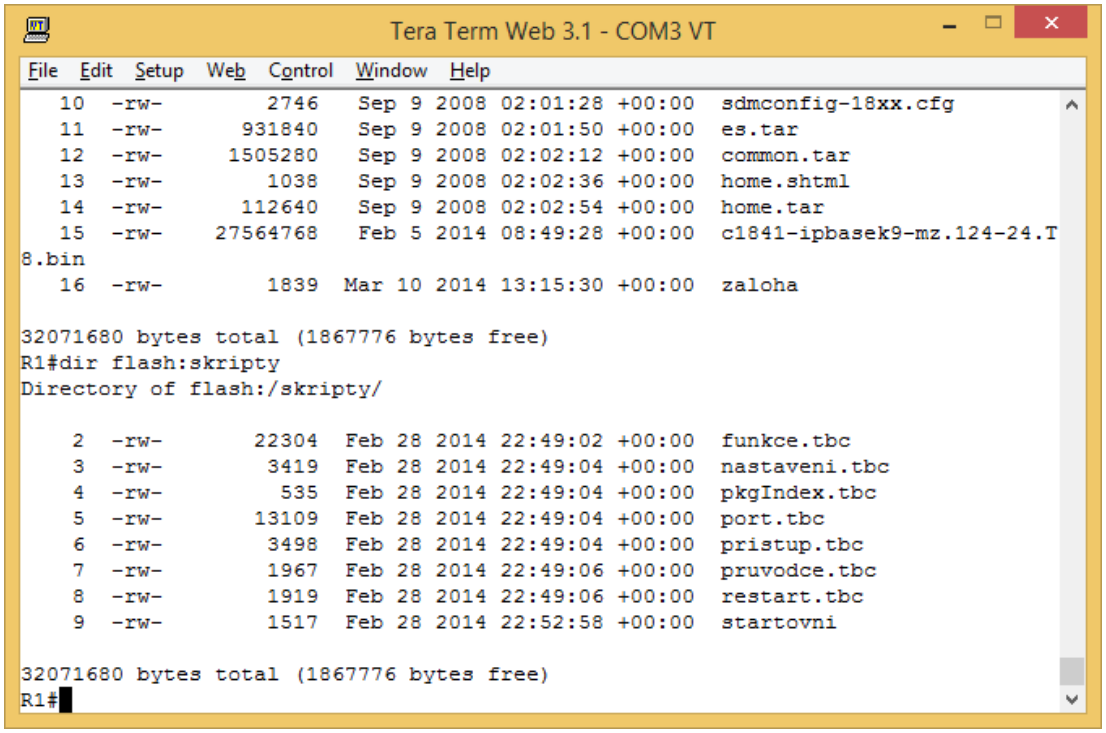


Obrázek 7.4: Ukázka konfigurace č. 4



Obrázek 7.5: Ukázka konfigurace č. 5

Poslední obrázek 7.6 ukazuje výpis adresáře „skripty“ v paměti prvku. Jedná se o kompilovanou verzi skriptů, které byly zkopírovány a rozbaleny pomocí skriptu „kopirovani.tbc“, který se sám na prvku nenachází.



Obrázek 7.6: Ukázka konfigurace č. 6

8. Testování

Testování probíhalo na prvcích uvedených v tabulce 8.1.

Prvek	**Model**	**Operační systém**	**Verze TCL**
směrovač	1841	12.4(24)T8 RELEASE SOFTWARE (fc1)	8.3.4
směrovač	2901	15.1(4)M4 RELEASE SOFTWARE (fc1)	8.3.4
přepínač	2960	12.2(55)SE5 RELEASE SOFTWARE (fc1)	8.3.4

Tabulka 8.1: Testovací prvky

Při testování skriptů bylo použito několika testovacích scénářů, a to především tak, aby byly prováděny různé konfigurační úlohy a docházelo tak ke kombinování částečných konfiguračních úloh, které budou pravděpodobně nejčastějším využitím. Ke konfiguraci všech částí, které průvodce umožňuje, bude pravděpodobně v jedné iteraci docházet méně často, ale i tato možnost byla otestována.

Jako částečnou konfiguraci lze označit např. nastavení jména prvku a vzdáleného přístupu k němu, ale již bez konfigurace dalších částí. Těchto částečných konfigurací lze využít především s postupujícím výkladem, kdy již probraná část je pro urychlení konfigurace nastavena právě pomocí průvodce, zatímco další části, které budou teprve probírány, jsou již dokonfigurovávány ručně uživatelem. Lze je však konfigurovat právě i pomocí průvodce při opakovaném spuštění, čímž dochází k jednotlivým iteracím průvodce.

Kompletní konfigurace při testování zahrnovala využití všech možností daných průvodcem, nikoli však „nekonečnou“ konfiguraci. Je tak zřejmé, že např. při konfiguraci interfaců, může uživatel konfigurovat neustále třeba i jeden interface stále dokola, tím zaplnit veškerou dostupnou paměť a docílit spadnutí skriptu. Nebylo by však příliš vhodné omezit početně konfiguraci jen např. na 10 iterací pro interface a 4 pro DHCP. Je tak ponechána volnost a možnost délky jednorázové konfigurace tak záleží především na aktuálně dostupné volné paměti prvku.

Testy probíhaly nejen po restartu prvku na instalačními skripty připravené konfiguraci, ale i jako opakované spouštění pomocí příkazu „pruvodce“ na již po-

změněné konfiguraci. Ke změně konfigurace prvku sloužily nejen předchozí iterace Konfiguračního průvodce, ale i změny provedené pomocí příkazů přímo uživatelem.

Byl proveden i uživatelský test vycházející z testovacích scénářů, na jehož základě byly upraveny některé části, které se týkají uživatelského rozhraní a byly tak odstraněny jeho drobné nedostatky.

Skripty prošly všemi provedenými testy a při dodržení doporučeného postupu jejich zavedení na prvek, který je uveden v manuálu (viz. Přílohy – Konfigurační průvodce manuál), jsou plně funkční.

9. Návrhy pro budoucí řešení

Jako vylepšení současných skriptů by se dalo implementovat odchytávání speciálních znaků, kdy pokud uživatel zadá chybný vstup a chce jej sám opravit, nedojde k jeho smazání, nýbrž vypsání speciálních znaků. Totéž se děje i při např. použití kurzorových šipek. Tyto sekvence znaků by se daly odchytávat a odstraňovat.

Dále by se dala implementovat optimalizace provádění konfiguračních příkazů tak, aby nedocházelo např. k zbytečnému zapínání interfacu, pokud je již zapnut a tím k zatěžování prvku.

Konfiguračního průvodce lze poměrně snadno rozšířit o další části, a to přidáním dalšího spouštěného skriptu do skriptu pruvodce.tcl.

U prvků, které mají málo operační paměti, by mohly skripty fungovat na principu okamžité konfigurace, kdy po zadání údajů uživatelem by docházelo ihned ke konfiguraci prvku a ne k ukládání do listu nebo by mohla být kontrolována dostupná volná paměť a podle ní by docházelo k úpravě možné celkové velikosti prováděné konfigurace, aby nedošlo k pádu skriptu.

10. Závěr

Hlavním cílem této bakalářské práce bylo vytvoření sady skriptů, které by usnadnily a urychlily opakovanou konfiguraci prvků v rámci výuky. K dosažení tohoto cíle bylo nutné provést analýzu, které části konfigurace prvků se při výuce opakují a navrhnout optimální řešení. Výsledkem této práce je tedy sada skriptů, která funguje jako Konfigurační průvodce a provede uživatele základním nastavením prvku. Hlavní dosaženou výhodou vyplývající z využívání těchto skriptů by mělo být zjednodušení a zrychlení opakující se základní konfigurace. Tím by mělo dojít k nárůstu volného času, který může být využit pro podrobnější a hlubší procvičení právě probírané problematiky. Jako další dosaženou výhodu použití těchto skriptů lze označit jednoduchost provádění konfiguračních změn, při kterých uživatel ani nemusí znát všechny potřebné konfigurační příkazy pro provedení požadovaných změn, ale je jen dotazován průvodcem na prováděné změny. V neposlední řadě je díky systému pokládání otázek zamezeno i opomenutí některých kroků, ke kterým by se jinak bylo třeba při konfiguraci vracet.

Dalšími cíli této práce bylo popsat skriptovací jazyk TCL, který byl využit pro psaní daných skriptů a to včetně zásadních příkazů, které byly využity. Dále nastínit funkci jednotlivých vytvořených skriptů, přiblížit technologie, které byly využity při psaní skriptů i ty, které umožňují použití přidaných funkcí a navrhnout možné využití vytvořených skriptů.

Skripty byly důkladně za použití několika testovacích scénářů otestovány na několika prvcích a vše je plně funkční. Všechny vytyčené cíle této bakalářské práce se tak dají označit za splněné.

Seznam použité literatury

- A through B: alias. *Cisco IOS Configuration Fundamentals Command Reference* [online]. © 1992–2013 [cit. 2014-03-19]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/ios/fundamentals/command/reference/cf_book/cf_a1.html#wp1013037.

- Array. *The Tcler's Wiki* [online]. 2013 [cit. 2014-03-13]. Dostupné z: http://wiki.tcl.tk/1032.

- BLAIR, Ray, Arvind DURAI a John LAUTMANN. *Tcl scripting for Cisco IOS*. Indianapolis, IN: Cisco Press, c2010, xvi, 294 p. Cisco Press networking technology series. ISBN 15-870-5945-2.

- Building reusable libraries – packages and namespaces. FLYNT, Clif. *Tcl Tutorial* [online]. [2003] [cit. 2014-03-10]. Dostupné z: http://www.tcl.tk/man/tcl8.5/tutorial/Tcl31.html.

- Cisco IOS Embedded Event Manager (EEM). *Management Instrumentation* [online]. © 1992–2013 [cit. 2014-03-13]. Dostupné z: http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-embedded-event-manager-eem/index.html.

- Cisco IOS Scripting with Tcl. *Cisco IOS Scripting with TCL Configuration Guide, Cisco IOS Release 12.4T* [online]. 2011 [cit. 2014-03-10]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ios_tcl/configuration/12-4t/ios-tcl-12-4t-book/nm-script-tcl.html.

- Command Scheduler (Kron). *Cisco Networking Services Configuration Guide, Cisco IOS XE Release 3S (Cisco ASR 1000)* [online]. © 1992–2013 [cit. 2014-03-13]. Dostupné z: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/cns/configuration/xe-3s/asr1000/cns-xe-3s-asr1000-book/cns-cmd-sched.html.

- Download TclPro 1.4. *Tcl Developer Xchange* [online]. 2001 [cit. 2014-03-10]. Dostupné z: http://www.tcl.tk/software/tclpro/eval/1.4.html.

- EEM System Architecture [obrázek]. *Cisco IOS Software Release 12.2SB New Features and Hardware Support, Product Bulletin 3258* [online]. 2008 [cit. 2014-03-10]. Dostupné z: `http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-software-releases-12-2-s/prod_bulletin0900aecd80586fe2.html`.
- OUSTERHOUT, John. History of Tcl. *Tcl Developer Xchange* [online]. 2001 [cit. 2013-06-05]. Dostupné z: `http://www.tcl.tk/about/history.html`.
- PEPELNJAK, Ivan. Kron: poor-man's cron. *IpSpace* [online]. 2007 [cit. 2014-03-13]. Dostupné z: `http://blog.ipspace.net/2007/11/kron-poor-man-cron.html`.
- PEPELNJAK, Ivan. Using Tcl packages on Cisco IOS. *IpSpace* [online]. 2007 [cit. 2014-03-10]. Dostupné z: `http://blog.ipspace.net/2007/09/using-tcl-packages-on-cisco-ios.html`.
- Securing Tool Command Language on Cisco IOS. *Security Intelligence Operations* [online]. © 1992–2013 [cit. 2014-03-10]. Dostupné z: `http://www.cisco.com/web/about/security/intelligence/securetcl.html`.
- Support and More. *Tcl Developer Xchange* [online]. 2001 [cit. 2013-06-05]. Dostupné z: `http://www.tcl.tk/about/support.html`.
- TIŠNOVSKÝ, Pavel. Programovací jazyk TCL. *ROOT.CZ* [online]. 2005 [cit. 2014-03-10]. Dostupné z: `http://www.root.cz/clanky/programovaci-jazyk-tcl/`.
- Uses for Tcl/Tk. *Tcl Developer Xchange* [online]. 2001 [cit. 2013-06-05]. Dostupné z: `http://www.tcl.tk/about/uses.html`.

Seznam tabulek

Seznam obrázků

Seznam použitých zkratek

EEM – Embeded Event Manager 13

IOS – Internetwork Operating System 3

MLS – Multilayer switch 26

TCL – Tool Command Language 3

Tk – Toolkit 11

Přílohy

Všechny přílohy jsou, vzhledem k jejich rozsahu a formě, umístěny na CD nosiči v elektronické podobě.

- **Dokumentace** – podrobný popis všech skriptů, složka dokumentace
- **Konfigurační průvodce manuál** – návod, jak zprovoznit skripty Konfiguračního průvodce na prvku, Konfigurační průvodce manuál.pdf
- **Skripty** – soubory skriptů. Jsou přiloženy obě verze a to již zabalené v .tar archivu:
  - *zkompilované* – složka skripty/zkompilované
  - *nezkompilované* – složka skripty/nezkompilované