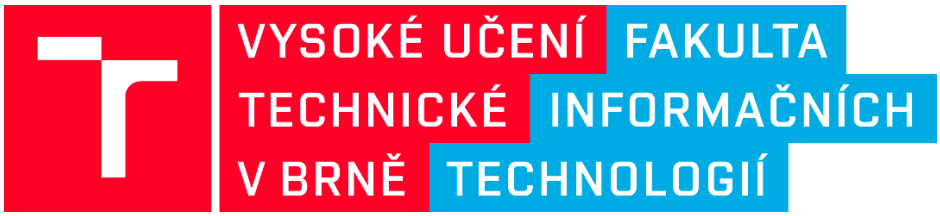
Vysoké učení technické v Brně

Fakulta informačních technologií



Síťové aplikace a správa sítí

Dokumentace LDAP serveru

Tomáš Aubrecht xaubre02

6.listopadu 2017

Obsah

1 Úvod

1.1 Lightweight Directory Access Protocol

Jedná se o protokol pro ukládání a přístup k datům na adresářovém serveru. Jednotlivé položky jsou na serveru ukládány formou záznamů a jsou uspořádány do stromové struktury. LDAP je odlehčenou verzí odvozenou od mezinárodního standardu X.500. Používá se pro práci s informacemi o uživatelích nebo udržování adresářů.

Komunikace serveru v projektu probíhá následovně:

- Klient zahájí komunikaci se serverem odesláním zprávy *BindRequest*.
- Server na ni odpoví zprávou *BindResponse*.
- Klient pomocí zprávy *SearchRequest* požádá o vyhledání záznamu.
- Server vyhledá požadované záznamy a pro každý záznam odešle samostatnou zprávu *SearchResEntry*.
- Po odeslání všech nalezených záznamů server informuje klienta o ukončení vyhledávání pomocí zprávy *SearchResDone*.
- Klient následně uzavře komunikaci zprávou *UnbindRequest*.

Každá zpráva je zapouzdřena do *LDAPMessage* představující datovou jednotku protokolu. Ta obsahuje identifikační číslo zprávy, o které se stará klient, a konkrétní typ zprávy. Více podrobností v oficiální specifikaci. Jednotlivé části zprávy jsou zakódovány pomocí ASN.1, což je prostředek pro popis datových struktur. Využívá metodu BER.

1.2 Basic Encoding Rules

Jedná se o soubor pravidel pro zakódování abstraktní informace do konkrétního datového (bytového) proudu. Každý datový prvek je zakódovaný jako trojice:

- Type – identifikátor typu
- Length – informace o délce
- Value – vlastní hodnota,

čímž se BER řadí mezi TLV formáty.

2 Struktura a implementace projektu

Projekt je rozdělen do tří hlavních částí: **argumenty, kodér/dekodér** a samotný **server**. Jednotlivé části bylo možné vypracovávat téměř nezávisle, což zároveň představovalo tři hlavní pomyslné milníky vývoje projektu, které byly potřeba zdolat.

V hlavní funkci programu se pracuje se dvěma objekty reprezentující parser pro argumenty a samotný server, který zapouzdřuje kodér a dekodér. Nejprve jsou zpracovány vstupní argumenty programu, kde pokud vše proběhlo úspěšně, tak jsou následně předány serveru. Server tak zahájí svou činnost a v případně úspěšné inicializace začne zpracovávat příchozí požadavky od klientů.

Důležitou roli zde hraje i funkce pro odchytávání signálů, která nastavuje příznak běhu programu, který je ukončen signálem *SIGINT* (klávesy CTRL + C).

2.1 Argumenty programu

Pro zpracování vstupních argumentů programu je zde implementovaná samostatná třída *„Params“* realizující tento proces. Při inicializaci této třídy jsou konstruktoru předány vstupní argumenty programu, které se následně zpracovávají.

Jako první proběhne kontrola počtu argumentů. Jsou zde pouze dvě možnosti, jak program spustit, a to s povinným přepínačem pro soubor a pak případně s volitelným přepínačem pro explicitní nastavení portu a jejich příslušné hodnoty. Dohromady včetně názvu programu to dává 3 nebo 5 argumentů. Jiný počet není povolen a program skončí s chybou.

Následně je provedena kontrola přepínačů, zdali je uveden povinný přepínač *„-f“* pro soubor, případně *„-p“* pro port, a uložení po nich následujících argumentů představující dané hodnoty.

V poslední části se provede kontrola uvedených hodnot. U souboru je zjišťuje, zdali se jedná o existující soubor a zdali se nejedná o adresář. U portu se zkontroluje, zdali se jedná o platné číslo v rozsahu 1 až 65535 (0 je rezervovaná a 65535 je maximální hodnota unsigned 16-bit int).

V případě výskytu jakékoliv chyby se nastaví chybový příznak, ke kterému lze přistoupit pomocí funkce *„valid()“,* a uloží se zpráva informující o chybě, která může být vytištěna na standartní chybový výstup pomocí funkce *„printError()*“.

2.2 Kodér a dekodér

Pro dekódování příchozích zpráv a kódování odchozích zpráv je zde implementováno několik tříd, které se o tuto činnost starají.

2.2.1 Třída IntegerBER

Dekóduje typ *Integer* a poskytuje jeho hodnotu, jeho délku v oktetech a jeho nedekódovanou verzi. Zkontroluje, zdali se opravdu jedná o typ *Integer* a zdali má správnou délku, kdy nemůže být na více oktetech, než daná zpráva obsahuje. Samotné dekódování je prováděno pomocí shiftování daných oktetů. Pokud je například délka *Integeru* 3 oktety, jeho hodnota je reprezentována binární posloupností těchto oktetů, proto je potřeba ve výsledku první oktet shiftovat doleva o 2 oktety, tudíž 16 bitů, druhý oktet shiftovat o 8 bitů a třetí se ponechá tak jak je.

2.2.2 Třída LengthBER

Pomocná třída pro zpracování délek jednotlivých typů. Poskytuje hodnotu délky a její délku v oktetech. Získávání hodnoty je stejné jako u *IntegerBER.*

2.2.3 Třída StrBER

Umožňuje zakódovat a dekódovat řetězce typu *OCTET STRING*. Poskytuje zakódovanou a dekódovanou hodnotu řetězce a celkovou délku v oktetech. Zakódování provede tím způsobem, že jako první oktet uvede typ OCTET STRING, druhý oktet délku daného řetězce a následující oktety jsou jednotlivé hodnoty znaků řetězce. Dekódování funguje podobně, ale jsou zde navíc kontroly typu a délky.

2.2.4 Třída PartialAttribute

Slouží pro vytvoření položky *PartialAttribute* v *PartialAttributeList* pro zprávu *SearchResultEntry*. Využívá k tomu výše zmíněnou třídu *StrBER*. *PartialAttribute* je sekvence řetězců udávajících typ a jeho hodnotu.

2.2.5 Třída LDAPFilter

Představuje filtr pro vyhledávání záznamů v CSV souboru. Poskytuje svůj typ, délku v oktetech, typ požadovaného atributu a jeho hodnoty, které můžou být buď *AssertionValue,* představující řetězec, se kterým se musí daná hodnota shodovat, řetězec, kterým musí daná hodnota začínat, pole řetězců, které jsou podřetězci dané hodnoty nebo řetězec, kterým má daná hodnota končit. Dále poskytuje pole filtrů pro operace AND, OR a NOT.

Dekódování probíhá tím způsobem, že se začíná na začátku filtru a postupuje se po jednotlivých bytech, kde se kontroluje jejich délka a syntaktická správnost dle oficiální specifikace a postupně se ukládají potřebné hodnoty. V případě výskytu jakékoliv chyby se patřičně nastaví návratová hodnota a dekódování se ukončí.

2.2.6 Třída MyLDAPMsgDecoder

Jednotný dekodér pro všechny podporované příchozí zprávy od klienta. Poskytuje typ dané zprávy, ID zprávy a návratový kód pro odpověď na tuto zprávu, *sizeLimit* představující maximální počet vyhledaných záznamů, seznam klientem vyžádaných atributů záznamů a filtr vyhledávání.

Při dekódování se postupuje stejným způsobem jako u filtru.

2.2.7 Třída MyLDAPMsgConstructor

Slouží pro vytváření odpovědí pro klienta. Umožňuje vytvoření *BindResponse*, *SearchResultEntry* a *SearchResultDone*, k čemuž využívá všech implementovaných kódovacích tříd. Každá tato zpráva je vytvořena na základě oficiální specifikace LDAP.

2.3 Server

Poslední zbývající částí je server, který umožňuje přijímání požadavků od klientů, jejich zpracování a následně odesílání adekvátních odpovědí. Server je v programu reprezentován třídou *„MyLDAP“*, která je inicializována odkazem na příznak běhu programu a cestou k souboru, se kterým se bude pracovat.

Pro zahájení činnosti je nejprve potřeba navázat server na daný port a začít naslouchat pro příchozí připojení. Pokud se v průběhu vyskytne nějaká chyba, program se ukončí a na standartní chybový výstup se vypíše odpovídající chybové hlášení.

Hlavní smyčka programu se nachází ve funkci *„serve()“,* kde se v cyklu, který je ukončen změnou příznaku běhu programu po přijetí signálu *SIGINT*, kontroluje, zdali se neobjevila nějaká žádost o připojení. Pokud ano, tak se server pokusí navázat spojení a v případě úspěchu se vytvoří nové vlákno zpracovávající dané připojení. V případě chyby při připojení se vypíše chybové hlášení a cyklus se opakuje. Pokud zde není žádné příchozí připojení, funkce na 100ms spí a následně se cyklus opakuje.

Zpracování jednotlivých požadavků je dáno funkcí *„process()“*, kde si server s klientem sekvenčně vyměňuje zprávy. Jako první server očekává zprávu *BindRequest,* kterou pomocí dekodéru dekóduje. Pokud se jedná o jinou zprávu než *BindRequest,* server vypíše hlášení a ukončí spojení, jinak odešle klientovi zprávu *BindResponse* s patřičným výsledným kódem.

Další zpráva, kterou server očekává, je v případě správného předchozího *BindRequestu SearchRequest,* pokud se vyskytla chyba, tak *UnbindRequest, kdy ukončí spojení, a* pro ostatní zprávy server vypíše chybové hlášení a ukončí spojení.

Pokud se jedná o *SearchRequest* a dekódování proběhlo úspěšně, server začíná s hledáním požadovaných záznamů. Nejprve otevře soubor a postupně prochází jednotlivé řádky, kde každý řádek vloží do konstruktoru struktury *„csv_record*“, která zkontroluje, zdali se jedná o platný CSV záznam a uloží si jednotlivé hodnoty. Záznam musí obsahovat 3 položky oddělené středníkem, kde první položka je jméno, druhá login a třetí je email. Pokud je záznam platný, aplikuje se na něj filtr dle požadavku klienta. Pokud je záznam neplatný, pokračuje se na další řádek.

Jakmile se projde celý soubor, začnou se výsledné záznamy odesílat klientovi zpět ve zprávách *SearchResultEntry*. Posílají se do doby, než jsou odeslány všechny záznamy nebo než není překročen *sizeLimit*. V tom případě se patřičně nastaví návratový kód. Pokud nebyl nalezen žádný záznam odpovídající filtru klienta, nemusí být poslána žádná zpráva *SearchResultEntry.*

Server ale vždy informuje klienta o ukončení vyhledávání pomocí zprávy *SearchResultDone*, která obsahuje návratový kód.

Následně server očekává pouze zprávu *UnbindRequest* pro ukončení spojení, ale vzhledem k tomu, že server už nepodporuje žádné jiné zprávy, tak po dokončení vyhledávání a přijetí další zprávy server ukončuje spojení bez ohledu na typ poslední zprávy.

2.3.1 Aplikace filtru equalityMatch

Podle zadaného atributu v *SearchRequest* se provede přímé porovnání s danou hodnotou CSV záznamu. Pokud daný záznam splňuje požadavek, je přidán do výsledného seznamu, jinak se pokračuje k dalšímu záznamu.

2.3.2 Aplikace filtru substrings

Podle zadaného atributu v *SearchRequest* se nejdříve zkontroluje, zdali je specifikován podřetězec, kterým má daná hodnota CSV záznamu začínat. Pokud ano, tak se zjistí, zdali je tento podřetězec podřetězcem dané hodnoty a následně zdali jím daná hodnota začíná.

Dalším krokem je kontrola všech podřetězců, pokud byly uvedeny, které se mohou vyskytovat na jakékoliv pozici, ale musí být v daném pořadí, v jakém byly zaslány. To je realizováno vyhledáváním podřetězců v dané hodnotě od pozice posledního nalezeného podřetězce. První podřetězec je hledán od začátku hodnoty.

Jako poslední se zkontroluje, zdali je specifikován podřetězec, kterým má daná hodnota CSV záznamu končit. Pokud ano, tak se zjistí, zdali je tento podřetězec podřetězcem dané hodnoty a následně zdali jím daná hodnota končí.

Pokud nejsou splněny všechny výše uvedené podmínky, tak je záznam nevyhovující a pokračuje se k dalšímu záznamu, jinak se uloží do výsledného seznamu.

2.3.3 Aplikace filtru AND

Rekurzivně se volá funkce na aplikaci filtru pro jednotlivé filtry, ze kterých je složena operace AND. Výsledné seznamy každého filtru jsou uloženy do pomocného seznamu, který se následně zpracuje. Z tohoto seznamu seznamů se vytáhne poslední seznam, prochází se jeho jednotlivé záznamy a kontroluje se, zdali je daný záznam obsažen ve všech ostatních seznamech. Pokud ano, operace AND je splněna a záznam se přidá do výsledného seznamu, jinak se přistupuje k dalšímu záznamu.

2.3.4 Aplikace filtru OR

Rekurzivně se volá funkce na aplikaci filtru pro jednotlivé filtry, ze kterých je složena operace OR. Výsledné záznamy každého filtru jsou uloženy do výsledného seznamu, ze kterého se později odstraní duplicitní záznamy.

2.3.5 Aplikace filtru NOT

Rekurzivně se volá funkce na aplikaci filtru pro daný filtr operace NOT, kde se výsledné záznamy ukládají do pomocného seznamu. Tento seznam se následně prochází a nevyskytuje-li se v něm aktuální záznam, tak se přidá do výsledného seznamu.

Rekurzivní volání umožňuje kombinace a zanořování jednotlivých filtrů.

3 Testování

Správné zpracování argumentů serveru bylo otestováno lokálně spouštěním programu s různými parametry.

Testování komunikace, kódování a dekódování bylo provedeno pomocí softwaru ldapsearch představující klientskou část a poskytnuté databáze obsahující CSV záznamy. Tento software je dostupným na univerzitním serveru Eva. Program byl spuštěn na univerzitním serveru Merlin a následně se na něj posílali klientské dotazy zahrnující veškeré možnosti nastavení filtrů a výstup se porovnával s předem vytvořeným referenčním výstupem.

4 Použití

Program se spouští v následujícím formátu: *$(./myldap -f [soubor] {-p <port>}),* kde

-f <soubor> je povinný argument, který představuje cestu k textovému souboru ve formátu CSV a *-p <port>* je volitelný argument, který umožňuje specifikaci portu, na kterém bude server naslouchat. Výchozí hodnota portu je 389.

5 Zdroje

- Zadání projektu:
  https://wis.fit.vutbr.cz/FIT/st/course-sl.php.cs?id=645445&item=64358&cpa=1

- Logo VUT FIT Brno:
  https://www.vutbr.cz/data_storage/multimedia/jvs/loga/02_fakulty/FIT/1-zakladni/CZ/PNG/FIT_barevne_RGB_CZ.png

- Vzorová databáze:
  http://nes.fit.vutbr.cz/ivesely/isa2017-ldap.csv

- LDAP specifikace:
  přednášky předmětu ISA, RFC 2254, RFC 4511
  https://cs.wikipedia.org/wiki/LDAP

- ASN.1 specifikace:
  https://cs.wikipedia.org/wiki/Basic_Encoding_Rules