Implementační dokumentace k projektu do ISA - LDAP Server

Autor: Filip Polomski, xpolom00
Datum: 20. 11. 2023

- Implementační dokumentace k projektu do ISA - LDAP Server
  - Cíl projektu
- Teorie
  - LDAP
    - Klien-server
    - TCP
  - Komunikace
  - Kódování zpráv - BER
    - BER
- LDAP server
  - Spouštění
  - Implementace
  - Testování pomocí ldapsearch
    - EQUALITY MATCH
    - SUBSTRING
    - OR
    - AND
- Zdroje

Cíl projektu

Cílem projektu bylo vytvořit síťovou aplikaci LDAP server podle protokolů pro LDAPv2.

Teorie

LDAP

Lightweight Directory Access Protocol (LDAP) je otevřený protokol, který je používaný k ukládání a načítání dat z hiearchické adresářové struktury. Jednotlivé položky jsou na serveru ukládány formou záznamů. Tento protokol se běžně používá pro ukládání

informací o organizacích a jejich uživatelých. U LDAP se využívá architektury klient-server a pro spolehlivé přenášení dat se používá TCP.

Klien-server

Klient-server je centralizovaná síťová architektura, kde probíhá komunikace mezi klientem a serverem na principu dotazování klientem pro služby, které jsou nabízeny serverem např. v tomto případě LDAP server.

TCP

Protokol transportní vrstvy, který zaručuje spolehlivý přenos dat ve spojení point-to-point (jeden odesílatel a jeden příjemce). Na začátku každé komunikace inicializuje odesílatel spojení s příjemcem. Následně pak může začít odesílat data. Po příchodu všech dat je příjemce zkontroluje podle kontrolního součtu a pokud bude vše v pořádku zašle odesílateli potvrzení o přijetí. Pokud ovšem bude odesílatel čekat na potvrzení příliš dlouho, nebo příjde požadavek o znovu zaslání, zašle data znovu (způsobeno např. ztrátou nebo zpožděním dat).

Komunikace

Komunikace serveru s klientem probíhá pomocí 6-ti základnách zpráv:

- *BindRequest* - zpráva, kterou klient zahajuje komunikace
- *BindResponse* - zpráva, kterou posílá server zpět klientovi po úspěšnám zpracování *BindResponse*
- *SearchRequest* - zpráva, kterou klient žádá server o vyhledávání dat
- *SearchResultEntry* - zpráva, kterou server odpovídá na dotaz *SearchRequest*, kde server odesílá klientovi jeden nalezený záznam
- *SearchResultDone* - zpráva, kterou server informuje klienta o dokončeném vyhledávání
- *UnbindRequest* - zpráva, kterou klient posílá serveru pro uzavření komunikace

Úspěšná komunikace pak může vypadat takto:

1. Klient pošle serveru *BindResponse* a tím zahájí komunikaci.
2. Server odešle *BindResponse* klientovi a tím potvrzuje navázanou komunikaci.
3. Klient odešle *SearchRequest*, který běžně obsahuje nějaký filter pro vyhledávání v databázi.
4. Servem následně odesílá klientovi postupně *SearchResultEntry* zprávy, které vždy obsahují jeden záznam.
5. Po odeslání všech záznamů server informuje klienta o dokončení hledání zprávou *SearchResultDone*.

6. Klient může poslat další *SearchRequest* zprávu nebo ukončí komunikaci se serverem zprávou *UnbindRequest*.

Kódování zpráv - BER

LDAP server používá pro kódování zpráv BER způsob

BER

Pro kódování informací používá BER formát *Tag-Length-Value* (Značka-délka-hodnota). Značka udává druh dat, které budou následovat, délka udává délku dat a hodnota pak reprezentuje skutečná data. Hodnota se pak může skládat z dalších *TLV* dat.

LDAP server

Jedná se o síťovou aplikaci obsahující LDAPv2 server pro vyhledávání dat ze souboru pomocí *ldapsearch*.

Spouštění

Server je možné spustit následujícím příkazem:

```
./isa-ldapserver {[-p <port> | --port <port>]} [-f <file> | --file <file>]
```

Kde jednotlivé argumenty znamenají:

- -p or --port "číslo portu". Výběr portu na kterém bude server naslouchat klientů. Výchozí hodnota čísla portu je 389.
- -f or --file "soubor". Cesta k textovému souboru formátu CSV.

Nebo lze soubor spustit:

```
./isa-ldapserver [-h | --help]
```

Pro vypsání používání aplikace.

Implementace

LDAPv2 server je implementováno v souborech v adresáři *src* s hlavičkovými soubory v adresáři *include*.

Nejprve je spouštěná funkce *main* v souboru *main.cpp*, který zakládá třídu *Server* ze souboru *server.cpp* a spouští její metodu pro parsování vstupních argumentů, kontrétně portu a souboru. Dále spustí metodu *run()* pro běh serveru.

Server vytváří třídu *Socket* ze souboru *socket.cpp*, kde spustí její metodu na *createSocket* pro vytvoření socketu a dále probíhá nastavení paralelního a neblokujícího serveru. A dále se zakládá třída *LDAP* ze souboru *LDAP.cpp* a spouští se její běh metodou *run()*.

Dále je zde vytvořena třída *ldapParser* ze souboru *ldapParser.cpp* u které se spouští metoda *msgParse()*, kde probíhá načítání zprávy do vectoru a zavoláním metodou *parseLDAPMessage* se zpracovájí dané zprávy posílané klientem ze sekce komunikace.

V metodě *parseLDAPMessage* se následně vytvářejí třídy *Sender* ze souboru *sender.cpp* a *DBreader* ze souboru *DBreader.cpp*.

Třída *Sender* se stará o posílání zpráv zpět kklientovi viz sekce komunikace. U *SearchRequest* zprávy probíha v metodě *parseSearchRequest()* zpracování zprávy, kde se volá důležitá metoda *loadFilters()*, která se stará o rekurzivná zpracování a dekódování filtru z *SearchRequest* zprávy. Po zpracování této zprávy se volá metoda třídy *DBreader* *run()* pro vyhledávání záznamů podle daného filtru a vrací vector záznamů, které jsou následně třídou *Sender* odesílány jako zprávy *SearchResultEntry* viz sekce komunikace.

Po odeslání všech záznamů se volá metoda třídy *Sender* *SearchResultDone* pro potvrzení dokončení hledání. A následně třída *ldapParser* čeká na další zprávu *SearchRequest* nebo *UnbindRequest*. Pokud příjde další požadavek na hledání, začne se nové hledání. Pokud příjde *UnbindRequest* server ukončuje komunikaci.

Testování pomocí ldapsearch

Testování probíhalo ručně a bylo kontrolováno s výstupy z ldapsearch ze serveru merlin. Níže jsou testovány základní filtry.

EQUALITY MATCH

INPUT

```
ldapsearch -x -H ldap://merlin:5063 "(uid=xpolom00)"
```

OUTPUT

```
# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
```

```
# filter: (uid=xpolom00)
# requesting: ALL
#

# xpolom00, fit.vutbr.cz
dn: uid=xpolom00,dc=fit,dc=vutbr,dc=cz
cn: Polomski Filip
uid: xpolom00
mail: xpolom00@stud.fit.vutbr.cz

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

SUBSTRING

INPUT

```
ldapsearch -x -H ldap://merlin:5063 "(uid=x*polom*00*)"
```

OUTPUT

```
# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
# filter: (uid=x*polom*00*)
# requesting: ALL
#

# xpolom00, fit.vutbr.cz
dn: uid=xpolom00,dc=fit,dc=vutbr,dc=cz
cn: Polomski Filip
uid: xpolom00
mail: xpolom00@stud.fit.vutbr.cz

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

OR

INPUT

```
ldapsearch -x -H ldap://merlin:5063 "(|(uid=xpolom00)(uid=xzelen27))"
```

OUTPUT

```
# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
# filter: (|(uid=xpolom00)(uid=xzelen27))
# requesting: ALL
#

# xpolom00, fit.vutbr.cz
dn: uid=xpolom00,dc=fit,dc=vutbr,dc=cz
cn: Polomski Filip
uid: xpolom00
mail: xpolom00@stud.fit.vutbr.cz

# xzelen27, fit.vutbr.cz
dn: uid=xzelen27,dc=fit,dc=vutbr,dc=cz
cn: Zelenak Martin
uid: xzelen27
mail: xzelen27@stud.fit.vutbr.cz

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

AND

INPUT

```
ldapsearch -x -H ldap://merlin:5063 "(&(uid=xpolom00)(cn=Polomski Filip))"
```

OUTPUT

```
# extended LDIF
#
# LDAPv3
# base <> (default) with scope subtree
# filter: (&(uid=xpolom00)(cn=Polomski Filip))
```

```
# requesting: ALL
#

# xpolom00, fit.vutbr.cz
dn: uid=xpolom00,dc=fit,dc=vutbr,dc=cz
cn: Polomski Filip
uid: xpolom00
mail: xpolom00@stud.fit.vutbr.cz

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Zdroje

1. Oracle. (bez data). *Understanding LDAP*. Načteno z https://docs.oracle.com/javase/jndi/tutorial/ldap/models/v2.html
2. Vysoké Učení Technické v Brně. (bez data). *Detail zadání projektu*. Načteno z https://www.vut.cz/studis/student.phtml?sn=zadani_detail&apid=268266&zid=54268
3. Wahl, M., Howes, T., & Kille, S. (1997). *Lightweight Directory Access Protocol (v3) (RFC 2251)*. Načteno z https://datatracker.ietf.org/doc/html/rfc2251#ref-3
4. OSS Nokalva, Inc. (bez data). *Basic Encoding Rules*. Načteno z https://www.oss.com/asn1/resources/asn1-made-simple/asn1-quick-reference/basic-encoding-rules.html#Tags
5. ITnetwork. (bez data). *C# Tutorial: TCP Připojení Komunikace*. Načteno z https://www.itnetwork.cz/csharp/sit/csharp-tutorial-tcp-pripojeni-komunikace/
6. Wikipedia. (bez data). *Transmission Control Protocol*. Načteno z https://cs.wikipedia.org/wiki/Transmission_Control_Protocol