VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Dokumentácia
Projekt - varianta Discord bot
ISA 2020/2021

Adam Múdry (xmudry01) 18. novembra 2020

Obsah

1 Úvod

Úlohou tohto projektu bolo naprogramovať program *isabot*, ktorý bude komunikovať so servermi služby Discord - zachytávať správy od používateľov (ak sa nachádzajú v kanáli s názvom *#isa-bot*) a odpovedať na ne vo formáte: "**echo:** <**username**> **-** <**message**>".

Zoznam odovzdaných súborov:

```
src/main.cpp src/main.hpp src/ssl.cpp src/ssl.hpp src/etc.cpp src/etc.hpp
Makefile
manual.pdf
README
```

1.1 Preklad

Program sa prekladá pomocou nástroja **make** spusteného v koreni priečinku:

```
make
```

alebo

```
make build
```

Pre vytvorenie ladiacej verzie programu použite príkaz:

```
make build-debug
```

Na vymazanie priečinku **tmp** s vytvorenými objektovými súbormi použite príkaz:

```
make clean
```

Makefile spúšťa program **g++** s nasledujúcimi parametrami:

```
--std=c++17 -Wall -Wpedantic [-O3|-g]
```

a linkuje vytvorené objektové súbory na knižnice pomocou:

```
-lcrypto -lssl
```

Vytvára sa spustiteľný súbor **isabot** na koreni priečinku.

1.2 Použitie

Príkaz na spustenie:

```
./isabot [-h|--help] [-v|--verbose] -t <bot_access_token>
```

```
Príznaky a argument:
    -h, --help                          Ukáže pomocnú hlášku
    -v, --verbose                       Zapne podrobný výpis
    -t, --token <bot_access_token>      Discord bot autentifikačný token
```

Pred použitím programu musí existovať jeho spustiteľný súbor.

2 Teória

Aby sa program *isabot* mohol spojiť a komunikovať so servermi služby Discord, musí vedieť používať technológiu **SSL**, **HTTPS** a vedieť sa autentifikovať (musí mať vygenerovaný **token**, ktorý preukáže identitu nášho programu). Po úspešnom zahájení bezpečnej komunikácie potom môže posielať HTTP **GET** a **POST** požiadavky smerujúce na **Discord API** [1], pomocou ktorých bude prebiehať komunikácia.

2.1 SSL

Secure Sockets Layer (doslova vrstva bezpečných socketov) je protokol, resp. vrstva vložená medzi vrstvu transportnú (napr. **TCP/IP**) a aplikačnú (napr. **HTTP**), ktorá poskytuje zabezpečení komunikácie šifrovaním a autentifikáciu komunikujúcich strán. [2]

2.2 HTTPS

Hypertext Transfer Protocol Secure je protokol umožňujúci zabezpečenú komunikáciu v počítačovej sieti. HTTPS využíva protokol HTTP spolu s protokolom *SSL* nebo *TLS*. HTTPS je využívaný predovšetkým pre komunikáciu webového prehliadača s webovým serverom. Zaisťuje autentifikáciu, dôvernosť prenášaných dat a ich integritu. Štandardný port na strane serveru je 443 TCP. [3]

2.3 Autentifikácia

Discord na autentifikáciu pri komunikácii s našim programom používa protokol **OAuth2**. Náš bot *isabot* má vygenerovaný token, ktorým preukáže svoju identitu a následne mu bude umožnená komunikácia s *Discord API*.

2.4 Komunikácia

Po vytvorení úspešného bezpečného spojenia a autentifikovania nášho bota nastáva komunikcácia s *Discord API* pomocou HTTP *GET* požiadaviek na získanie informácii o skupinách (**Guilds**) a kanáloch (**Channels**), na ktoré je tento bot napojený a následné načítanie správ z kanálov s názvom *#isa-bot* a HTTP *POST* požiadaviek, pomocou ktorých bot odosiela správy naspäť na server.

Ukážka HTTP *GET* požiadavky pre získanie správ z určitého kanálu:

```
GET /api/channels/<ID kanálu>/messages HTTP/1.1\r\n
Host: discord.com\r\n
Authorization: Bot <Token>\r\n
\r\n
```

Ukážka HTTP *POST* požiadavky pre odoslanie odpovedi na server:

```
POST /api/channels/<ID kanálu>/messages HTTP/1.1\r\n
Host: discord.com\r\n
Authorization: Bot <Token>\r\n
Content-Type: application/json\r\n
Content-Length: <Dĺžka správy>\r\n
\r\n
{"content": "echo: <Meno používateľa> - <Správa>"}\r\n
\r\n
```

3 Implementácia

Zadanie projektu som implementoval v jazyku C++ písaného v štandarde C++17.

3.1 Načítavanie argumentov

Po spustení programu sa načítavajú argumenty zo štandardného vstupu (*stdin*) pomocou funkcie

```
void parseArgs(int& argc, char* argv[], bool& show_help, bool& verbose, std::string&
    token);
```

ktorá využíva knižnicu *getopt.h* a jednotlivé načítané argumenty vráti do premenných z parametrov funkcie.

3.2 Zahájenie SSL spojenia

Po načítaní *tokenu* a ostatných argumentov zo štandardného vstupu program naviaže SSL spojenie pomocou funkcie

```
int initConnection(SSL** ssl_return);
```

ktorá vracia ukazovateľ na štruktúru **SSL** využíva OpenSSL knižnice *openssl/ssl.h*, *openssl/err.h* a pár iných knižníc určených pre programovanie so sieťami ako napr. *netinet/in.h*. [4]

3.3 Získavanie informácii

Získanú *SSL* štruktúru môžeme využiť v nasledujúcich funkciách

```
int sendPacket(SSL* ssl, const char *buf);
std::string recieveResponse(SSL* ssl);
```

z ktorých prvá posiela naše *GET* alebo *POST* požiadavky na Discord API a druhá spracováva odpoveď a vráti ju vo forme *std::string*. [4]

Telo HTTP odpovede je vo formáte **JSON**, ktorý program spracuje pomocou funkcií zo štandardnej knižnice C++ **regex** a naplní štruktúry

```
struct Message; // id, user_id, channel_id, timestamp, username, content, ...
struct Channel; // id, guild_id, name, messages, ...
struct Guild;   // id, name, channels, ...
```

ktoré obsahujú dáta ako id, názvy a podobne a sú vzájomne napojené.

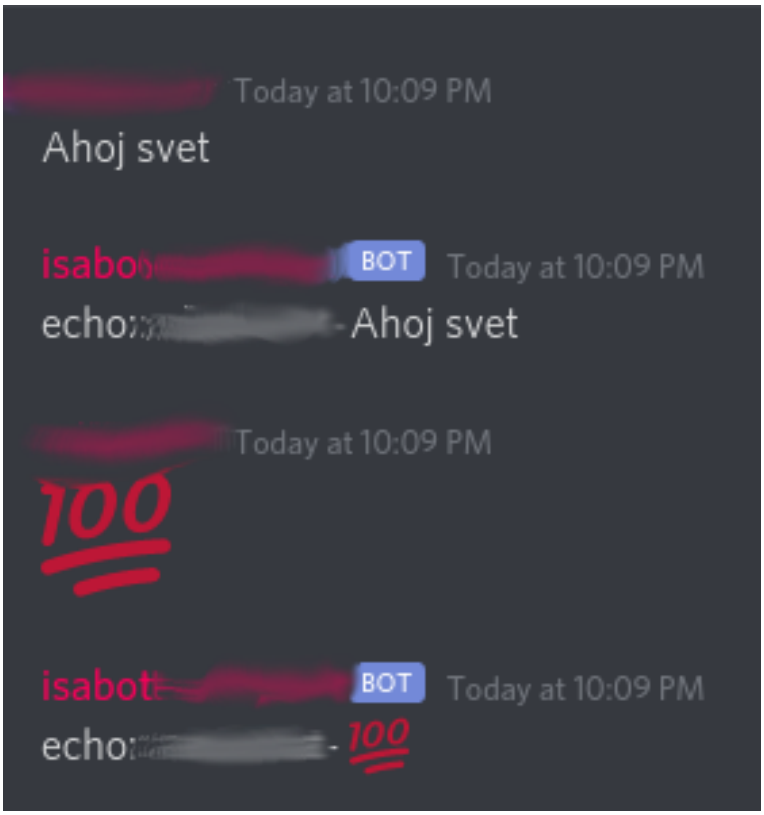
3.4 Odpovedanie na správy používateľov

Po načítaní všetkých potrebných informácii môže program začať načítavať správy z jednotlivých kanálov a odpovedať na ne. Toto sa deje v jednom **while** cykle. Program porovnáva lokálny čas a čas odoslania správy, získaný z *JSON* odpovede Discord serveru. Ak je správa staršia ako porovnaný čas, program ju ignoruje - ak je však správa nová, program ju zaregistruje a pošle odpoveď na server. Program taktiež ignoruje správy, ktoré poslali iný boti.

Isabot odpovedá s krátkym oneskorením, aby nepresiahol limit HTTP požiadaviek určený z Discord API.

4 Testovanie

Program som testoval manuálne pomocou 2 vlastných skupín a kanálov a vlastného Discord účtu.



5 Bibliografia

Literatúra

[1] Discord Developer Documentation. [online].
URL https://discord.com/developers/docs

[2] Secure Sockets Layer. [online].
URL https://cs.wikipedia.org/wiki/Secure_Sockets_Layer

[3] HTTPS. [online].
URL https://cs.wikipedia.org/wiki/HTTPS

[4] O.logN: Stack Overflow. [online].
URL https://stackoverflow.com/questions/41229601/openssl-in-c-socket-connection-https-client