Síťové aplikace a správa sítí
Projekt – Discord bot

Michal Rein (xreinm00)

20.10.2020

Obsah

1 Úvod do problematiky

Cílem je vytvoření programu pro zachytávání zpráv na Discord serveru, jejich následné zpracování a odeslání zpět na daný server. V praxi podobné programy operující nad servery a vykonávající nějakou automatizaci označujeme jako boti. Součástí zadání je využití SSL socketů z knihoven openssl. Využejeme tedy REST API rozhraní, které nám poskytuje Discord pro přístup k serverům a pomocí zabezpečené HTTPS komunikace TLS verze 1.2 budeme průběžně sbírat data a odesílat je zpět na server.

2 Základní informace

2.1 Soubory projektu

- main.cpp
  - Soubor se zdrojovým kódém, implementace třídy ISABot.
- main.h
  - Hlavičkový soubor k souboru main.cpp, deklarace třídy ISABot.
- message.cpp
  - Soubor se zdrojovým kódem, implementace třídy MessageProcessor.
- message.h
  - Hlavičkový soubor k souboru message.cpp, deklarace třídy MessageProcessor a datové struktury Message.
- Makefile
  - Soubor pro sestavení programu pomocí příkazu ‚make‘.

2.2 Použití aplikace

Aplikace se přeloží příkazem *make*, přičemž vznikne spustitelný soubor *isabot*. Program lze spuštět s těmito parametry:

- **-h | --help:**
  - Vypíše nápovědu na standardní výstup
- **-v | --verbose:**
  - Bude zobrazovat zprávy, na které bot reaguje na standardní výstup ve formátu <channel> - <username>: <message>

- **-t <bot_access_token>:**
  - Autentizační token pro přístup bota na Discord

Parametr **-t** s tokenem pro bota je povinný, ostatní volitelné.

3 Implementace

Aplikace byla napsána v programovacím jazyce C++. Podrobnější popisy funkcí včetně parametrů a návratových hodnot lze nalézt v hlavičkovém souboru „main.h“ a „message.h“.

3.1 Použité knihovny

- <openssl/ssl.h, err.h>
  - Poskytuje komunikaci pomocí SSL
- <netdb.h>
  - Metody pro překlad IP adres na DNS adresu.
- <arpa/inet.h>
  - Struktury a metody pro práci s IP adresami.
- <getopt.h>
  - Zpracování vstupních argumentů.
- <iostream, iomanip>
  - I/O výpisy a manipulátory.
- <sys/socket.h, netinet/in.h>
  - Metody poskytující sockety a práci s nimi.

3.2 Chybové stavy

Program generuje chybové stavy a vrací návratové kódy podle následujícího rozdělení:

- ARGUMENTS_ERROR
  - Chyby vstupních argumentů
  - Ukončení s kódem 1
- CONNECTION_ERROR
  - Chyby při vytváření nebo ukončení spojení
  - Ukončení s kódem 2
- PARSING_ERROR
  - Chyby při zpracování obsahu zpráv
  - Ukončení s kódem 3

Většina zjištěných chyb vypisuje také chybovou hlášku na standardní chybový výstup.

3.3 Popis implementace

Program je logicky rozdělen do 2 částí: třída **ISABot** a třída **MessageProcessor**, přičemž tato třída slouží jako doplňek funkcionality pro hlavní třídu ISABot.

Po spuštění aplikace je vytvořena instance třídy ISABot a následně zavolána metoda *run()* nad tímto vzniklým objektem. Samotná metoda slouží k inicializaci všech nezbytných částí bota, získání informací o serverech a vyhledání kanálu pro zachytávání zpráv. Program se nejdříve snaží získat informace o tom, na které servery má přístup pomocí metody *get_guild_id()*. Po úspěšném nalezení serveru je potřeba nalézt kanál s označením „isa-bot“, o který žádá pomocí metody *get_channels()*. Poslední potřebnou informací je identifikátor poslední zprávy na kanále, která je získána metodou *get_last_message_id()*. Následně se program zacyklí do smyčky pro opakované zachytávání komunikace a případné odesílání zpráv zpět na server.

Pro jednoznačnou identifikaci registrovaného bota v aplikaci Discord je využíván tzv. ***token***, který obsahuje unikátní řetězec identifikující bota v rámci celé soustavy serverů, které spadají pod aplikaci Discord. K tomuto tokenu jsou pak vázana nejrůznější práva a přístup k jednotlivým serverům. Tento řetězec je potřeba vygenerovat v aplikaci Discord a je nezbytný pro fungování celého programu.

Pro komunikaci se serverem používá bot metody *connect_to_server(), send_request(), recv_request()* a *close_connection(),* vždy v tomto pořadí, ať už se jedná o požadavek GET nebo POST. V prvním kroku se program musí pokusit navázat spojení se serverem pomocí metody *connect_to_server()*. Tato metoda nejdříve inicializuje síťový socket pomocí metody *init_socket()*, a následně vytvoří SSL spojení se serverem metodou *init_ssl()*.

Jakmile je spojení úspěšné, je možné vyvolat libovolný požadavek pomocí metody *send_request()*. Ta vyžaduje parametry *variant (int)* a *message (string)*. Parametr *variant* určuje jednu z předdefinovaných zpráv, označenou enumerátorem jako jednu z následujících:

- GET_MESSAGES
  - Vrátí všechny nové zprávy na kanále
- SEND_MESSAGE
  - Odešle zprávu na server danou parametrem *message (string)*
- GET_GUILDS
  - Vrací seznam serverů (Discordem interně označované jako Guilds), ke kterým má bot s daným tokenem přístup
- GET_CHANNELS
  - Vrací kánály přítomné na daném serveru (identifikátor serveru se nachází v privátní proměnné guilds)
- GET_LAST_MESSAGE
  - Vrací identifikátor poslední zprávy na kanále

Po odeslání požadavku je potřeba zavolat metodu *recv_response()*, která se postará o přijetí odpovědi od serveru, a jestliže je nastaven parametr *parse (bool)* na hodnotu True, odpověď serveru bude předána **instanci třídy MessageProcessor**, konkrétně přes její metodu *parse_messages()*. Výstupem této metody je list datových struktur *Message*, který reprezentuje a uchovává všechny nalezené zprávy. Z tohoto listu poté bot čerpá jednotlivé zprávy a na základě nastavení příznaku *bot (bool)*, který udává, zda zpráva byla vygenerována nějakým jiným botem (nebo sebou samým), odesílá obsah zprávy zpět na server.

Bot poté opakovaně odesílá zprávy GET_MESSAGES a zpracovává přijaté odpovědi ve smyčce. Každá iterace obsahuje zpozdění (konkrétně 1s), aby nedocházelo k zahlcení serveru, který má poté tendenci bota ignorovat a odepřít mu přístup k informacím.

Identifikátor každé zprávy je zaznamenán do vnitřní proměnné bota, a je použit jako hodnota parametru *after*, díky které server vrací pouze zrávy, které se objevily až chronologicky po poslední zpracované zprávě.