



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

IRC BOT S LOGOVÁNÍM SYSLOG

IRC BOT WITH SYSLOG LOGGING

SÍŤOVÉ APLIKACE A SPRÁVA SÍTÍ

NETWORK APPLICATIONS AND NETWORK ADMINISTRATION

AUTOR

AUTHOR

MARTINA GRZYBOWSKÁ

BRNO 2017

Obsah

1	Úvod	2
2	Dôležité pojmy	3
2.1	IRC protokol	3
2.1.1	Formát správ IRC protokolu	3
2.2	Syslog protokol	4
2.2.1	Formát správ Syslog protokolu	4
3	Návrh aplikácie	5
4	Popis implementácie	6
4.1	Spracovanie vstupných argumentov	6
4.2	Vytvorenie spojenia s IRC serverom	6
4.3	Interpretácia prichádzajúcich IRC správ	6
4.4	Odosielanie správ Syslog serveru	7
4.5	Špeciálne funkcie bota	7
4.5.1	Funkcionalita ?today	7
4.5.2	Funkcionalita ?msg	7
5	Testovanie	10
6	Informácie o programe	11
6.1	Návod na použitie	11
7	Záver	12
	Literatúra	13

Kapitola 1

Úvod

Cieľom tejto dokumentácie je popísať riešenie projektu pre predmet Sieťové aplikácie a správa sietí. Našou úlohou je vytvoriť a otestovať jednoduchého IRC bota s logovaním Syslog v jazyku C/C++. Tento bot sa pomocou knižnice BSD sockets pripája na IRC server a odchytáva správy, ktoré následne analyzuje, triedi a odosiela vo formáte Syslog protokolu na Syslog server.

Kapitola 2 tejto dokumentácie obsahuje stručný úvod do základných pojmov protokolov IRC a Syslog. V kapitole objasňujeme návrh našej aplikácie, zatiaľ čo samotná implementácia je popísaná v kapitole . Kapitola je zamerná na testovanie a ladenie aplikácie v priebehu riešenia zadania. Kapitola uvádza základné informácie o výslednej aplikácii a návod na jej použitie.

Kapitola 2

Dôležité pojmy

V tejto kapitole predstavujeme úvod do problematiky IRC a Syslog protokolov.

2.1 IRC protokol

IRC protokol je definovaný v RFC 1459 [2]. Jedná sa o protokol aplikačnej vrstvy, ktorý je založený na klient-server architektúre. Jeho hlavnou úlohou je efektívne sprostredkovať textovú komunikáciu v reálnom čase. Bol vyvinutý na systémoch využívajúcich TCP/IP sieťový protokol.

Hlavnou oporou IRC protokolu je IRC server. Poskytuje bod, ku ktorému sa klienti a ďalšie servery môžu pripájať, čím efektívne vytvára IRC sieť. Užívatelia sa na server pripájajú pomocou IRC klienta. Každý užívateľ je od ostatných rozlišovaný pomocou jedinečnej prezývky, ktorej maximálna dĺžka je deväť znakov. Užívatelia majú možnosť zoskupovať sa do kanálov, teda pomenovaných skupín, ktoré existujú do doby, kým je na nich prihlásený aspoň jeden užívateľ. Užívatelia môžu nadobúdať funkciu operátora kanálu, pričom takýto užívateľ je označený znakom @.

Kanály môžu byť dvoch rôznych typov, ktoré rozlišujeme podľa špeciálneho začiatočného znaku v ich pomenovaní. Prvým je distribuovaný kanál, označený znakom #, ktorý je viditeľný pre všetky servery pripojené do siete, a teda sa naň môže pripojiť každý užívateľ prihlásený na ktorýkoľvek server zo siete. Druhý typ kanálu, ktorý je označený špeciálnym znakom &, je viditeľný len pre server, na ktorom bol vytvorený, a teda sa naň môžu prihlásiť len užívatelia prihlásení na daný server. Po odoslaní správy v rámci kanálu správuvidia všetci užívatelia momentálne prihlásení na tento kanál.

2.1.1 Formát správ IRC protokolu

Syntax správ IRC protokolu je delená do troch hlavných častí. Prvou, nepovinnou, je komponenta <prefix>, nasledovaná už povinnou časťou <command>. Táto časť správy nesie informáciu o vykonávanom príkaze, ktorý môže byť niekoľkých typov, napríklad JOIN, NICK, PART, QUIT alebo KICK. Za časťou <command> nasleduje zoznamom parametrov <params>, ktorý môžeme ďalej rozložiť do komponent <middle> alebo <trailing>. Za <params> musí nasledovať povinné CRLF. Celá syntax formátu IRC správ je popísaná pomocou BNF reprezentácie, ktorej časť je zobrazená na obrázku 2.1. Zvyšné časti BNF reprezentácie, s ktorými pre účely našej aplikácie nebolo potrebné pracovať, sú popísané v kapitole 2.3.1 v dokumente RFC 1459 [1].

```

<message> ::= [ ':' <prefix> <SPACE> ] <command> <params> <crLf>
<prefix>  ::= <servername> | <nick> [ '!' <user> ] [ '@' <host> ]
<command> ::= <letter> { <letter> } | <number> <number> <number>
<SPACE>   ::= ' ' { ' ' }
<params>   ::= <SPACE> [ ':' <trailing> | <middle> <params> ]

<middle>   ::= <Any *non-empty* sequence of octets not including SPACE
               or NUL or CR or LF, the first of which may not be ':'>
<trailing> ::= <Any, possibly *empty*, sequence of octets not including
               NUL or CR or LF>

<crLf>     ::= CR LF

```

Obr. 2.1: BNF reprezentácia syntaxe IRC správ

2.2 Syslog protokol

Syslog protokol je definovaný v RFC 3164 [1]. Tento protokol poskytuje zariadeniu možnosť posielat logovacie správy naprieč sieťou ku kolektorom logovacích správ, tzv. syslog serverom. Syslog protokol na transportnej vrstve využíva UDP protokol s portom 514 alebo TCP protokol s portom 6514.

2.2.1 Formát správ Syslog protokolu

Správy Syslog protokolu sa rozdeľujú do troch častí, pričom dĺžka správy nesmie presiahnuť 1024 bajtov. Minimálna dĺžka siete špecifikovaná nie je, ale posielat Syslog správy bez obsahu by nemalo byť podporované.

Prvou časťou správy je časť PRI, ktorá je ohraničená znakom < a končí znakom >. Hodnota čísla PRI sa nazýva Priority (priorita) a je vypočítaná na základe vzorca

$$\text{Facility} * 8 + \text{Severity}$$

Facility reprezentuje zariadenie a Severity mieru závažnosti.

Druhou časťou je časť HEADER, ktorá obsahuje časovú značku TIMESTAMP a HOSTNAME, teda hostiteľské meno alebo IP adresu. Táto časť musí nasledovať ihneď za ukončujúcim znakom časti PRI, pričom za TIMESTAMP a HOSTNAME musí nasledovať vždy jedna medzera.

Poslednou časťou je časť MSG. Obsahuje pole TAG reprezentujúce názov procesu, ktorý túto správu vygeneroval a pole CONTENT, ktoré predstavuje text odoslanej správy. Za touto časťou už nenasleduje žiaden ukončujúci znak.

Celková syntax Syslog správy má teda podobu:

```
<PRI>TIMESTAMP HOSTNAME TAG CONTENT
```

Kapitola 3

Návrh aplikácie

Aplikácia začína načítaním vstupných argumentov príkazového riadku a ich prerozdelením do členov príslušných štruktúr. Následne je inicializované spojenie s IRC serverom a začína sa prijímanie správ. Každá prijatá správa je podrobená analýze, pri ktorej sa určuje typ správy spolu so všetkými informáciami potrebnými pre jej spracovanie. Pri prijatí istého druhu správy je spustená jedna zo špeciálnych funkcionalít našej aplikácie, teda vo výsledku je na IRC server odoslaná správa obsahujúca buď výpis aktuálneho dátumu (bližšie špecifikované v podkapitole 4.5.1), alebo meno odosielateľa a text pôvodne prijatej správy (bližšie špecifikované v podkapitole 4.5.2). V prípade prijatia správy vyhovujúcej zadaným kľúčovým slovám z argumentu `-1` aplikácia odosiela správu na Syslog server. Aplikácia beží dovtedy, než je ukončená signálom `SIGINT`, alebo kým nenastane nejaká chyba obmedzujúca funkcionalitu.

Kapitola 4

Popis implementácie

Aplikácia je implementovaná v jazyku C/C++. Nie je navrhnutá objektovo, ale využíva niektoré objekty zo štandardných knižníc jazyka C++. Pracuje s adresami typu IPv4. Pre prehľadnosť kódu je rozdelená do štyroch modulov a jedného hlavičkového súboru. Funkcia `main` je implementovaná v súbore `isabot.cpp`, zatiaľ čo funkcie pre spracovanie argumentov z príkazového riadku sa nachádzajú v module `arguments.cpp`. Modul `networking.cpp` sa stará o inicializácie spojení so servermi a odosielanie správ. V module `commands.cpp` sa nachádzajú funkcie na rozbor správ a hlavná logika aplikácie.

4.1 Spracovanie vstupných argumentov

Aplikácia má podľa zadania prijímať niekoľko vstupných argumentov. Pravidlá o počte výskytov voliteľných argumentov neboli bližšie špecifikované, a teda naša implementácia počíta s tým, že každý voliteľný argument sa môže vyskytnúť maximálne raz. Spracovanie vstupných argumentov a ich ukladanie do jednotlivých štruktúr je realizované vo funkcii `parse_args()`. V prípade akejkoľvek chyby je aplikácia ukončená chybovým kódom `ARGUMENTS_ERR`. Pokiaľ všetko prebehne bezchybne, kontext je prepnutý naspäť do funkcie `main`.

4.2 Vytvorenie spojenia s IRC serverom

Inicializácia spojenia s TCP IRC serverom je implementovaná vo funkcii `connect_to_tcp()`. V prípade akejkoľvek chyby aplikácia končí s chybovým kódom `TCP_CONNECTION_ERR`. Po nadviazaní spojenia so serverom aplikácia odošle tri úvodné správy `NICK`, `USER` a `JOIN`. Následne je zavolaná funkcia `receive_from_server()`, ktorá prijíma všetkú komunikáciu zo strany servera až do ukončenia aplikácie.

4.3 Interpretácia príchodzích IRC správ

Po prijatí správy z IRC serveru funkciou `receive_from_server()` je kontext prepnutý do funkcie `determine_command()`. Táto funkcia určí, o aký typ správy sa jedná, vyberie z nej všetky potrebné informácie a podľa typu zavolá príslušnú funkciu na vykonanie danej akcie. Jedná sa teda o funkciu implementujúcu hlavnú logiku programu.

4.4 Odosielanie správ Syslog serveru

Vyhľadávanie správ obsahujúcich kľúčové slová špecifikované vo vstupnom argumente `-l` je implementované vo funkcii `find_argument_keywords()`. Pokiaľ je nájdená zhoda, obsah tejto správy spolu s prezývkou užívateľa je odovzdaný funkcii `create_syslog_message()`. Táto funkcia vytvára správu podľa Syslog protokolu, štruktúra tejto správy je bližšie uvedená v podkapitole 2.2.1. V našom prípade máme stanovené, že zariadenie má byť vždy nastavené na `local0` (Facility, hodnota 16) a závažnosť je `Informational` (Severity, hodnota 6), teda podľa vzorca spomenutého opäť v podkapitole 2.2.1 dostávame hodnotu `PRI` konštatne ako 134. Následne zistíme aktuálny čas a naformátujeme ho do potrebnej podoby. Aktuálnu IP adresu zistíme pomocou funkcie `get_current_ip()`, čím vyplníme pole `HEADER`. Nasleduje časť `MSG`, pričom `TAG` sme získali už pri načítaní argumentov a `CONTENT` je tvorený užívateľským menom a obsahom správy pôvodne určenej na logovanie. Takto vytvorenú správu odovzdáme funkcii `send_syslog_message()`, ktorá správu odošle.

4.5 Špeciálne funkcie bota

Pri detekovaní správy `PRIVMSG` alebo `NOTICE` so špeciálnym kľúčovým slovom nachádzajúcim sa v časti `<trailing>`, má náš program vykonať jednu z funkcií, ktorých implementácia je popísaná v nasledujúcich podkapitolách. Detekcia správ obsahujúcich takéto slová je vykonávaná vo funkcii `find_special_keywords()`.

4.5.1 Funkcionalita `?today`

Po potvrdení, že `<trailing>` danej správy sa zhoduje s regulárnym výrazom

`^\\?today`

zistíme náš aktuálny dátum a vo formáte `dd.mm.yyyy` ho odošleme na kanál, z ktorého pôvodná správa prišla.

4.5.2 Funkcionalita `?msg`

Po potvrdení, že `<trailing>` danej správy sa zhoduje s regulárnym výrazom

`^\\?msg \\S+:.+`

má náš program za úlohu časť správy nachádzajúcu sa za `?msg` buď odoslať, pokiaľ je užívateľ v nej špecifikovaný práve prítomný na danom kanáli, alebo si ju odložiť a odoslať ju až vtedy, keď sa daný užívateľ prihlási na daný kanál. Bolo teda potrebné vymyslieť nie len mechanizmus na ukladanie správ priradených jednotlivým užívateľom a kanálom, ale taktiež spôsob, ako zisťovať či je daný užívateľ prítomný, alebo nie.

Pre ukladanie správ určených pre práve neprítomných užívateľov sme sa rozhodli využiť usporiadaný asociatívny kontajner `std::map`, ktorý obsahuje páry kľúč-hodnota, pričom kľúč je jedinečný. Naším kľúčom je prezývka užívateľa a hodnotou je ďalší kontajner `std::map`. Kľúčom tohto zanoreného kontajnera je názov kanálu a hodnotou je sekvenčný

kontainer `std::vector`, obsahujúci texty správ na odoslanie.

Pre zistenie prítomnosti užívateľa, pre ktorého je správa `?msg` adresovaná, je možné odoslať serveru správu s príkazom `NAMES`. Server následne odošle odpoveď obsahujúcu zoznam mien aktívnych užívateľov na danom kanále. Vzhľadom na to, že takéto preposielanie správ by sa muselo vykonať pri detekcii každej jednej `?msg` správy, sme sa rozhodli implementovať funkcionality inak - vytvorili si vlastnú databázu užívateľov a kanálov. Tento spôsob je síce implementačne náročnejší, ale žiadne nadbytočné preposielanie správ pri ňom neprebíha.

Pre realizáciu nášho spôsobu sme vytvorili ďalší usporiadaný asociatívny kontainer `std::map`, v ktorom je kľúčom opäť prezývka užívateľa, podobne ako v prvom prípade, avšak hodnotou je sekvenčný kontainer `std::vector`, obsahujúci názvy kanálov, na ktorých je daný užívateľ práve prítomný. Zachytávaním správ od serveru obsahujúcich príkazy `JOIN`, `NICK`, `PART`, `QUIT`, `KICK` a `RPL_NAMREPLY` (číslo 353), ich následným rozborom a uložením dát do príslušných kontainerov, dokážeme našu databázu pravidelne aktualizovať. Pre túto funkcionality sme špeciálne vytvorili sériu funkcií `command`, ktoré vo svojej podstate zrkadlia fungovanie IRC príkazov. Jedná sa o nasledujúce funkcie:

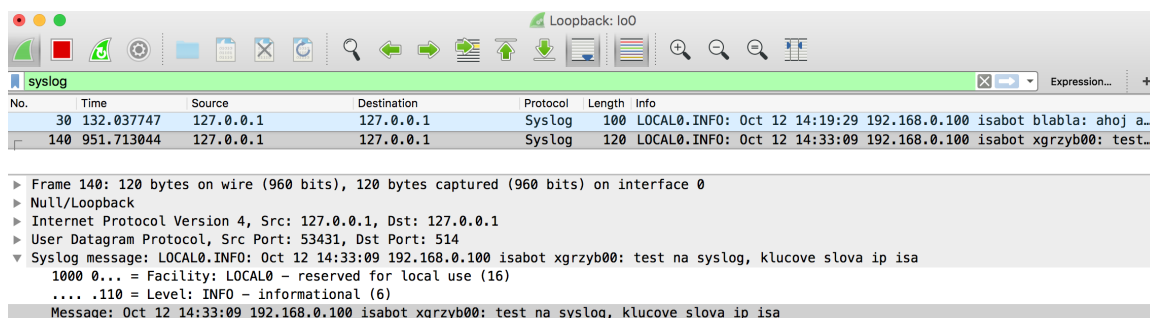
- Funkcia `command_names` sa stará o správu `RPL_NAMREPLY` (353), ktorá prichádza samostatne od každého kanálu ihneď po prihlásení. V časti `<trailing>` obsahuje menný zoznam aktívnych užívateľov v dobe nášho prihlásenia. Funkcia vkladá užívateľské prezývky do kontainera prítomných užívateľov a kanálov.
- Funkcia `command_join`, starajúca sa o príkaz `JOIN`, splňuje podobnú úlohu ako `command_names`, avšak navyše obsahuje volanie funkcie `find_pending_messages()`, ktorá zisťuje či novoprihlásený užívateľ má v kontaineri neprítomných záznam o neodoslaných správach, ktoré mu je potrebné odoslať. Pokiaľ sa nájde zhoda, správy sú odoslané a záznam o nich je odstránený. Pokiaľ daný užívateľ po tomto úkone už nemá žiadne ďalšie záznamy o správach na odoslanie pochádzajúcich z iných kanálov, je z kontainera neprítomných užívateľov odstránený.
- Funkcia `command_nick` je volaná v prípade obdržania príkazu `NICK`, ktorý ohlasuje zmenu užívateľskej prezývky. Keďže v zadaní našej úlohy nebolo bližšie špecifikované, za akých podmienok sa má aktivovať funkcionality `?msg`, v našej implementácii predpokladáme, že okrem prihlásenia sa na server ju môže spustiť aj príkaz na zmenu prezývky.
- Funkcia `command_quit` sa stará o príkaz `QUIT`, ktorý ohlasuje úplny odchod užívateľa zo serveru, a teda podobne aj naša funkcia vymaže užívateľa z kontainera prítomných užívateľov.
- Funkcia `command_part`, spravujúca príkaz `PART`, ohlasuje odchod užívateľa z jedného alebo viacerých kanálov. Pokiaľ užívateľ odchádza z rovnakého počtu kanálov, aký má zaregistrovaný, plní podobnú úlohu ako funkcia `command_quit`.
- Funkcia `command_kick` je volaná pri príchode príkazu `KICK`. Tento príkaz slúži k vyhodneniu užívateľa z kanála. V prípade, že je prezývka vyhodneného užívateľa zhoduje s prezývkou bota, program odošle správu `QUIT` a končí, pretože nie je schopný plnej funkcionality. Pokiaľ sa prezývka nezhoduje, len sa zavolá funkcia `command_part` ako keby sa užívateľ rozhodol z kanálu odísť sám.

Pre celkové objasnenie nášho riešenia ?msg funkcionality teraz zhrnieme všetky definované pojmy. Keď je detekovaná správa ?msg, funkcia `check_if_online()` skontroluje asociatívny kontajner obsahujúci užívateľov a kanály, na ktorých sú aktívni. Pokiaľ sa v ňom príjemca nachádza a je aktívny na kanáli, z ktorého bola správa odoslaná, ihneď sa mu prepošle. V opačnom prípade je takýto užívateľ spolu s informáciami o správe pridaný do asociatívneho kontajnera obsahujúceho neprítomných užívateľov. Vďaka funkciám zo série `command`, ktoré aktualizujú databázu pri všetkých odchodoch a príchodoch užívateľov, náš program okamžite zistí, ak sa užívateľ, ktorý má záznam o nevybavených správach, prihlási a uložené správy odošle.

Kapitola 5

Testovanie

Testovanie našej aplikácie prebiehalo prevažne na stránke webchat.freenode.net¹ pomocou voľne dostupného Python IRC bota a výpisových funkcií. Spočívalo v prihlásení sa na rovnaký kanál na stránke tromi rôznymi spôsobmi; našou aplikáciou, botom a ručne. Následným vykonávaním všetkých nami sledovaných príkazov výpisová funkcia postupne zobrazovala prijaté správy, stavy našich kontajnerov a text správ odosielaných na Syslog server. Testovanie správnosti formátu týchto Syslog správ prebiehalo pomocou nástroju Wireshark². Aplikácia bola testovaná na CentOS 6.5 a OS X 10.11.2, na oboch operačných systémoch je plne funkčná.



No.	Time	Source	Destination	Protocol	Length	Info
30	132.037747	127.0.0.1	127.0.0.1	Syslog	100	LOCAL0.INFO: Oct 12 14:19:29 192.168.0.100 isabot blabla: ahoj a...
140	951.713044	127.0.0.1	127.0.0.1	Syslog	120	LOCAL0.INFO: Oct 12 14:33:09 192.168.0.100 isabot xgrzyb00: test...

► Frame 140: 120 bytes on wire (960 bits), 120 bytes captured (960 bits) on interface 0

► Null/Loopback

► Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

► User Datagram Protocol, Src Port: 53431, Dst Port: 514

▼ Syslog message: LOCAL0.INFO: Oct 12 14:33:09 192.168.0.100 isabot xgrzyb00: test na syslog, klucove slova ip isa

1000 0... = Facility: LOCAL0 - reserved for local use (16)

.... .110 = Level: INFO - informational (6)

Message: Oct 12 14:33:09 192.168.0.100 isabot xgrzyb00: test na syslog, klucove slova ip isa

Obr. 5.1: Kontrola Syslog správ pomocou Wiresharku

¹<https://webchat.freenode.net/>

²<https://www.wireshark.org/>

Kapitola 6

Informácie o programe

Aplikácia je prekladaná pomocou Makefile. Skladá sa z nasledujúcich súborov:

- isabot.h
- isabot.cpp
- arguments.cpp
- networking.cpp
- commands.cpp

6.1 Návod na použitie

Aplikácia je spúšťaná cez terminál, pre správne spustenie je nutné buď zadať dva povinné vstupné argumenty, pričom zvyšné argumenty sú voliteľné, alebo zadať argument `-h`, respektíve `--help`. V prípade akejkoľvek chyby je aplikácia ukončená príslušným chybovým kódom a chybovou hláškou bližšie špecifikujúcou chybu. Užívateľ aplikáciu ukončuje odoslaním príkazu `SIGINT`.

Spôsob použitia:

```
isabot HOST[:PORT] CHANNELS [-s Syslog_SERVER] [-l HIGHLIGHT] [-h | --help]
```

Vysvetlenie významu jednotlivých argumentov:

```
HOST - názov serveru (napríklad irc.freenode.net)
PORT - číslo portu, na ktorom server naslúcha (predvolený 6667)
CHANNELS - názov jedného či viac kanálov, na ktoré se klient pripojí
-s Syslog_SERVER - IP adresa logovacieho Syslog serveru
-l HIGHLIGHT - zoznam kľúčových slov oddelených čiarkou
```

Kapitola 7

Záver

Cieľom tejto dokumentácie bolo zhrnúť a objasniť riešenie nášho projektu pre predmet Sieťové aplikácie a správa sietí. Pri riešení projektu sme mali možnosť vyskúšať a precvičiť si ako prácu s BSD sockets a sieťové programovanie, tak aj programovanie v C++ samotnom, keďže s týmto jazykom sme nemali veľa predošlých skúseností. Oboznámili sme sa s protokolmi IRC a Syslog a nadobudli cenné skúsenosti ohľadom získavania informácií z RFC súborov.

Literatúra

- [1] Lonvick, C.: The BSD syslog Protocol. RFC 3164, RFC Editor, Aug 2001.
URL <http://www.rfc-editor.org/rfc/rfc3164.txt>
- [2] Oikarinen, J.; Reed, D.: Internet Relay Chat Protocol. RFC 1459, RFC Editor, May 1993.
URL <http://www.rfc-editor.org/rfc/rfc1459.txt>