

Programowanie Sieciowe

# **Serwer do obsługi usługi IRC**

## **Z51. Demonstracja działania**

Maciej Bogusławski (331362)

Hubert Kaczyński (331386)

Bartosz Żelazko (331457)

## Spis treści

Wstęp.....	3
Prezentacja działania .....	3
Otwarcie połączenia klient- serwer .....	3
Wysyłanie wiadomości pomiędzy dwoma klientami .....	5
Stworzenie kanału .....	5
Dołączenie do kanału .....	6
Wysyłanie wiadomości do kanału .....	7
Opuszczenie kanału .....	8
Usunięcie użytkownika z kanału .....	8
Zakończenie połączenia .....	9

## Wstęp

W tym dokumencie znajduje się opis działania funkcjonalności związanych z działaniem serwera IRC w ramach projektu z przedmiotu programowanie sieciowe. Ze względu na fakt, iż prace naszego zespołu udało się przyspieszyć ukazane zostało działanie wszystkich funkcji opisanych w dokumentacji wstępnej projektu.

W celu wizualizacji działania użyto narzędzia weechat które jest zewnętrzną implementacją klienta IRC. Dodatkowo w tym dokumencie znajdują się zrzuty ekranu terminala, na których widoczne są logi generowane przez bibliotekę spdlog.

## Prezentacja działania

Aby zaprezentować działanie programu postanowiliśmy posłużyć się przypadkami użycia przedstawionymi w punkcie pod tytułem „Przypadki użycia” w dokumentacji wstępnej. W celu uzyskania większej czytelności w logach z serwera uwzględniono jedynie logi z poziomem info i wyższym.

## Otwarcie połączenia klient- serwer

Na samym początku pracy z naszym programem uruchamiany jest serwer IRC, który po uruchomieniu loguje fakt, iż rozpoczął on pracę oraz wypisuje na terminal parametry z jakimi pracuje. Poniżej zamieszczony został zrzut ekranu przedstawiający elementy wypisane na terminal chwilę po uruchomieniu serwera IRC.

```
❖ barzel@DESKTOP-VCAU1C8:~/PSI/repozytorium_elka/psi_z51_projekt_serwer_irc/irc-server$ ./bin/irc_server
[2026-01-04 01:50:05] [info] IRC server started

=====
IRC Server started!
Port: 6667
Waiting for connections
=====
```

Następnie w ramach demonstracji uruchomiony został program weechat, aby móc rozpocząć pracę z serwerem. Na samym początku pracy należy skonfigurować parametry takie jak adres IP i port na jakim działa serwer, wyłączenie TLS oraz ustawienie hasła dla serwera. Przykładowy proces konfiguracji został zaprezentowany na poniższym zrzucie ekranu.

```
23:44:51   irc: server added: test -> 0.0.0.0/6697 (TLS: enabled)
23:45:16   Option changed: irc.server.test.tls = off (default if null: on)
23:45:49   Option changed: irc.server.test.password = "123456" (default if null: "")
```

Po dokonaniu wstępnej konfiguracji w narzędziu weechat możemy zainicjalizować połączenie z serwerem IRC poprzez wykonanie polecenia `/connect <nazwa serwera>` w weechat. Komenda ta powoduje nawiązanie połączenia poprzez wysłanie wiadomości CAP PASS USER oraz NICK. Działanie tej komendy z poziomu weechat pokazuje zdjęcie na następnej stronie.

```

script, spell, trigger, typing, xfer
23:41:34 test === ===== End of backlog (20 lines) =====
23:41:34 test -- irc: connecting to server 0.0.0.0/6667...
23:41:34 test -- irc: connected to 0.0.0.0/6667 (0.0.0.0)
23:41:34 test -- irc: client capability, server supports:
23:41:34 test -- irc: client capability, requesting:
23:41:34 test -- Welcome to the IRC Network barzel!barzel@localhost
23:41:34 test -- Your host is myircserver.local, running version 0.1
23:41:34 test -- This server was created just now
23:41:34 test -- myircserver.local 0.1 o o
23:41:34 test -- - myircserver.local Message of the day -
23:41:34 test -- - Welcome to the our Z51 IRC server!
23:41:34 test -- End of /MOTD command.
[23:41] [1] [irc/test] 1:server[test]
[barzel]

```

Od strony serwera w przypadku takiego połączenia logi wyglądają w sposób ukazany poniżej.

```

[2026-01-03 23:41:34] [info] New client connected: 127.0.0.1:59140. Socket: 6
[2026-01-03 23:41:34] [info] Client 6 is asking about server capabilities
[2026-01-03 23:41:34] [info] Socket 6 nick set to: barzel
[2026-01-03 23:41:34] [info] Socket 6 set username to: barzel
[2026-01-03 23:41:34] [info] Socket 6 set realname to: barzel
[2026-01-03 23:41:34] [info] Client 6 has ended capability negotiation
[2026-01-03 23:41:34] [info] Client 6 registered with nick: barzel
[2026-01-03 23:42:34] [info] Responded to PING from client 6 with PONG

```

W przypadku w którym występuje kolizja nazw użytkowników i/lub nicków serwer podczas próby zalogowania odpowie odpowiednim błędem. Błąd ten jest poprawnie interpretowany przez klienta weechat, a on wyświetla użytkownikowi następujące wiadomości.

```

00:22:05 test -- irc: connected to 0.0.0.0/6667 (0.0.0.0)
00:22:05 test -- irc: client capability, server supports:
00:22:05 test -- irc: client capability, requesting:
00:22:05 test -- irc: nickname "barzel" is already in use, trying nickname "barzel2"
00:22:05 test -- Welcome to the IRC Network barzel2!nowy@localhost
00:22:05 test -- Your host is myircserver.local, running version 0.1
00:22:05 test -- This server was created just now
00:22:05 test -- myircserver.local 0.1 o o
00:22:05 test -- - myircserver.local Message of the day -
00:22:05 test -- - Welcome to the our Z51 IRC server!
00:22:05 test -- End of /MOTD command.
00:22:05 [1] [irc/test] 1:server[test]

```

Warto zwrócić uwagę na fakt, iż weechat, jako dość zaawansowany klient, po otrzymaniu błędu mówiącego o nieoryginalnym nickname sam proponuje nowy nickname. Jak widać na powyższym zrzucie ekranu, zaproponowany nowy nickname już jest oryginalny i udało się połączyć z serwerem. Od strony serwera w takiej sytuacji logowane są przedstawione powyżej zdarzenia.

## Wysyłanie wiadomości pomiędzy dwoma klientami

Po poprawnym połączeniu się z serwerem możemy wysyłać wiadomości prywatne pomiędzy użytkownikami. Na samym początku, aby dowiedzieć się, jacy użytkownicy znajdują się już w sieci, możemy skorzystać z komendy `/who`. Komenda ta służy do wyszukiwania użytkowników. Przykładowy wynik użycia tej komendy z poziomu klienta weechat znajduje się poniżej.

```
23:48:50 test -- [*] barzel (barzel@localhost) H 0 (barzel)
23:48:50 test -- [*] barzel2 (nowy@localhost) H 0 (nowy)
23:48:50 test -- [*] End of WHO list
```

W trakcie obsługi tego zapytania serwer generuje następujące logi.

```
[2026-01-03 23:48:50] [info] Sent WHO entry for client 6 to client 7
[2026-01-03 23:48:50] [info] Sent WHO entry for client 7 to client 7
[2026-01-03 23:48:50] [info] Sent WHO list to client 7
```

Wiedząc już jacy klienci znajdują się już w naszej sieci, korzystając z komendy `/msg <użytkownik> <treść wiadomości>` możemy wysłać komunikat PRIVMSG, który skutkuje wysłaniem wiadomości prywatnej do wyspecyfikowanego użytkownika. Klient wysyłający wiadomości zobaczy następujący komunikat w weechat.

```
23:50:00 test -- Msg(barzel2) -> barzel: witaj
```

A odbiorca zobaczy następujące powiadomienie:

```
1. test      nowy@localhost
   weechat   23:50:00 barzel2 | witaj
2. barzel2
```

Obsługa PRIVMSG przez serwer generuje następujący log:

```
[2026-01-03 23:50:00] [info] Delivered PRIVMSG from client 7 to barzel
```

## Stworzenie kanału

Aby stworzyć kanał należy wykonać operację `/join` gdzie jako argument podana jest nazwa nieistniejącego kanału. Wysłanie takiego komunikatu do serwera spowoduje stworzenie nowego kanału a osoba tworząca z automatu zostanie administratorem tego kanału. W weechat ten fakt jest symbolizowany przez znak `@` przed nickiem użytkownika. Dodatkowo po stworzeniu kanału możemy skorzystać z komendy `/topic` aby nadać temat dla danego kanału. Wykonanie obu tych komend widziane z poziomu klienta zostało zaprezentowane na następnej stronie.

```

1.test      mojTemat
  weechat    23:55:46 --> barzel (barzel@localhost) has joined #nowyKanal
2. barzel2   23:55:46 --  No topic set for channel #nowyKanal
3. #nowyKanal 23:55:46 --  Channel #nowyKanal: 1 nick (1 op, 0 voiced, 0 regular)
              23:56:49 --  barzel has changed topic for #nowyKanal to "mojTemat"

```

Operacje te po stronie serwera pozostawiają następujące logi.

```

[2026-01-03 23:55:46] [info] Responded to PING from client 6 with PONG
[2026-01-03 23:55:46] [info] Created channel: #nowyKanal, operator socket: 6
[2026-01-03 23:55:46] [info] Client barzel created and joined new channel #nowyKanal (socket 6)
[2026-01-03 23:56:22] [info] Responded to PING from client 7 with PONG
[2026-01-03 23:56:34] [info] Responded to PING from client 6 with PONG
[2026-01-03 23:56:49] [info] [TOPIC] barzel set topic in #nowyKanal to: mojTemat
[2026-01-03 23:57:22] [info] Responded to PING from client 7 with PONG
[2026-01-03 23:57:34] [info] Responded to PING from client 6 with PONG
[2026-01-03 23:58:22] [info] Responded to PING from client 7 with PONG

```

## Dołączenie do kanału

Aby otrzymać listę kanałów dostępnych na serwerze należy wysłać wiadomość LIST. Na tą wiadomość serwer powinien odpowiedzieć wysyłając listę wszystkich kanałów wraz z ich tematami. Przykładowa wizualizacja takiej listy wygenerowana z poziomu weechat znajduje się poniżej.

```

1.test      1 channels (total: 1) | Filter: * | Sort: ~name2 | Key(input): ctrl+j=join channel, ($) =refresh, (q)=close buffer
  weechat    #nowyKanal      1 mojTemat
2. list_test

```

Następnie możemy dołączyć do kanału poprzez wysłanie wiadomości JOIN. Po dołączeniu do kanału użytkownik powinien zobaczyć następujące informacje.

```

1.test      mojTemat
  weechat    23:59:26 --> barzel2 (nowy@localhost) has joined #nowyKanal
2. list_test 23:59:26 --  Topic for #nowyKanal is "mojTemat"
3. #nowyKanal 23:59:26 --  Channel #nowyKanal: 2 nicks (1 op, 0 voiced, 1 regular)

```

## Wysyłanie wiadomości do kanału

Wysyłanie wiadomości na kanał odbywa się poprzez wysłanie PRIVMSG gdzie zamiast nazwy użytkownika podawana jest nazwa kanału. Przykład wysłania wiadomości użytkowników na kanał został zamieszczony poniżej.

```
-- Sun, 04 Jan 2026 --
00:00:14 barzel2 | witam
00:00:22 @barzel | czesc
00:02:39 @barzel | to ja z nowym nickiem

[00:03] [3] [irc/test] 3:#nowyKanal{2}
[@barzel] |
```

Serwer natomiast w takiej sytuacji powinien wygenerować następujące logi:

```
[2026-01-04 00:00:14] [info] Delivered PRIVMSG from client 7 to channel #nowyKanal (1 members)
[2026-01-04 00:00:22] [info] Delivered PRIVMSG from client 6 to channel #nowyKanal (1 members)
```

Na tym etapie postanowiliśmy także zaprezentować jak wygląda zmiana nazwy użytkownika widocznej dla innych. Aby ją zmienić należy wykonać wysłać wiadomość NICK. Wysłanie takiej wiadomości zostanie zalogowane przez serwer w następujący sposób.

```
[2026-01-04 00:01:49] [info] Socket 6 nick set to: nowyNick
```

Po zmianie nicku i wysłaniu wiadomości inni użytkownicy zobaczą zmianę nicków.

```
00:00:14 barzel2 | witam
00:00:22 @barzel | czesc
00:02:39 nowyNick | to ja z nowym nickiem
```

## Opuszczenie kanału

Celem opuszczenia kanału należy wysłać do serwera wiadomość PART jako argument podając jako argument nazwę kanału z którego chcemy wyjść. W takiej sytuacji klient weechat powinien zaprezentować następujący komunikat.

```
00:04:08 <-- | barzel2 (nowy@localhost) has left #nowyKanal (WeeChat 4.1.1)
```

Serwer powinien zalogować następującą informację:

```
[2026-01-04 00:04:08] [info] [PART] Client barzel2 is leaving channel #nowyKanal - reason: WeeChat 4.1.1  
[2026-01-04 00:04:08] [info] Socket 7 left channel #nowyKanal
```

Po opuszczeniu kanału nie możemy już wysyłać w nim wiadomości. Próba wysłania wiadomości do serwera, który został przez nas już opuszczony skończy się zwróceniem przez serwer odpowiedniego błędu. Błąd ten weechat wyświetla w sposób następujący:

```
00:04:08 <-- | barzel2 (nowy@localhost) has left #nowyKanal (WeeChat 4.1.1)  
-----  
00:06:09 barzel2 | abc  
00:06:09 -- | #nowyKanal: Cannot send to channel
```

W takim przypadku serwer stworzy następujący log:

```
[2026-01-04 00:06:09] [warning] PRIVMSG command from client 7: not in channel #nowyKanal
```

## Usunięcie użytkownika z kanału

Aby usunąć użytkownika z kanału administrator może wykorzystać komendę */kick*. Wysyła ona do serwera wiadomość o tej samej nazwie która w przypadku podania odpowiednich parametrów poskutkuje usunięciem użytkownika. Udana wykonanie tej komendy zostało zaprezentowane poniżej.

```
00:21:30 <-- | nowyNick has kicked barzel2 (nowyNick)
```

W tym przypadku w nawiasie przedstawiony jest nick administratora, który zdecydował o wyrzuceniu użytkownika. W takiej sytuacji serwer generuje poniższe logi.

```
[2026-01-04 00:21:30] [info] [KICK] nowyNick is kicking barzel2 from #nowyKanal - reason: nowyNick  
[2026-01-04 00:21:30] [info] Socket 6 kicked socket 7 from channel #nowyKanal
```

W przypadku gdy użytkownik bez praw administratora postanowi wykonać komendę *kick*, serwer nie pozwoli mu na to i odeśle kod błędu który jest poprawnie interpretowany przez weechat.

```
00:11:44 -- | #nowyKanal: You're not channel operator
```

Serwer natomiast w takiej sytuacji stworzy następujący log:

```
[2026-01-04 00:11:44] [info] [KICK] Client barzel2 is not channel operator in #nowyKanal
```



Podobnie sprawa będzie wyglądać gdy administrator poda niepoprawną lub nieistniejącą nazwę użytkownika, który ma zostać wyrzucony. W weechat wygląda to następująco:

```
00:19:58 test -- | nowyNick: nieistnieje No such nick/channel
```

Server:

```
[2026-01-04 00:19:58] [info] [KICK] User does not exist: nieistnieje
```

Podobnie jak w przypadku wyjścia po wyrzuceniu użytkownik nie może wysyłać wiadomości do kanału po tym jak został wyrzucony.

```
00:21:30 <-- | nowyNick has kicked barzel2 (nowyNick)
00:22:53 barzel2 abcd
00:22:53 -- | #nowyKanal: Cannot send to channel

[00:22] [2] [irc/test] 2:(#nowyKanal){0} [H: 1]
[barzel2]
```

## Zakończenie połączenia

Aby zakończyć połączenie z serwerem należy wysłać wiadomość QUIT. Wysłanie wiadomości tej skutkuje także usunięciem użytkownika ze wszystkich kanałów, których był członkiem lub administratorem. Dodatkowo gdy liczba użytkowników kanału spadnie do zera serwer automatycznie usuwa dany kanał. Zakończenie połączenia i następujące po nim wyjście i usunięcie pustego kanału zostało zaprezentowane na poniższych zrzutach ekranu.

Przed quit:

```
2 channels (total: 2) | Filter: * | Sort: ~name2 | Key(input): ctrl+j=join channel, ($) =refresh, (q)=close buffer
#nowyKanal 1 No topic
#samotnykanal 1 No topic

1. test
weechat 00:25:18 --> | barzel2 (nowy@localhost) has joined #samotnyKanal @barzel2
2. #nowyKanal 00:25:18 -- | No topic set for channel #samotnyKanal
3. #samotnyKanal 00:25:18 -- | Channel #samotnyKanal: 1 nick (1 op, 0 voiced, 0 regular)

[00:25] [3] [irc/test] 3:#samotnyKanal{1} [H: 1]
[@barzel2] /quit
```

Po quit:

```
1.test          1 channels (total: 1) | Filter: * | Sort: ~name2 | Key(input): ctrl+j=join channel, ($) =refresh, (q)=close buffer
weechat        #nowyKanal      1 No topic
2. #nowyKanal
3. list_test

[01:20] [3] [irc/test] 3:server[test] [H: 1]
[barzel12]
```

Logi serwera:

```
[2026-01-04 01:19:42] [info] Client 6 sent QUIT: WeeChat 4.1.1
[2026-01-04 01:19:42] [info] Forcing disconnection of client 6 (nick barzel)
[2026-01-04 01:19:42] [info] Reason for disconnection: WeeChat 4.1.1
[2026-01-04 01:19:42] [info] Socket 6 left channel #samotnykanal
[2026-01-04 01:19:42] [info] Removing channel: #samotnykanal
[2026-01-04 01:19:42] [info] Removing client with socket 6
```