情報ネットワーク特論 レポート

25GJK05 佐藤 力斗

1. はじめに

私は、複数人が参加できるチャットのような仕組みを実装した。

2. 概要

このシステムでは、最初にクライアントがユーザ名をサーバに送信し、サーバ側でユーザ名の重複がないかどうかのチェックを行う。重複なしの場合、チャットに参加することができる。クライアントがテキストを送信すると、サーバがそれを他のクライアントに中継する。クライアントが接続または切断するたびに、サーバは他のクライアントへ通知を行う。なお、一度使用したユーザ名は再度利用できない。

3. プログラムの詳細

server.rbとclient.rbのプログラムの説明をそれぞれ行う。

3.1. server.rb

clientsにはクライアントの一覧、usernamesにはユーザ名を、used_usernameには過去のユーザ名を入れる。これらに情報を入れることでユーザ名の重複を防止する。

表1: 情報を入れる配列とハッシュのプログラム

```
clients = [] # クライアント一覧
usernames = {} # ユーザ名の対応表
used_usernames = [] # 過去のユーザ名
```

ユーザ名の重複があるかどうかの確認のプログラムを表2に示す。ユーザ名が使われている場合は、クライアント側をcloseすることで同じユーザ名が使われないようにしている。

表2: ユーザの重複確認のプログラム

```
    # ユーザ名の重複確認
    # 重複 → 強制的に終了
    if used_usernames.include?(username)
      client.puts "ERROR: そのユーザ名はすでに使われています"
      clients.delete(client)
      client.close
      next
    end
```

切断したことをサーバ側で表示し、クライアントに通知するプログラムを表3に示す。”ensure”のところにプログラムを書くことで、プログラムの最後に実行されるため退出したことを通知できるようにしている。

表3: 退出を通知するプログラム

```
    ensure # 最後に実行 (切断)
      clients.delete(client)
      name = usernames.delete(client)
      puts "#{name} が切断しました"

      # 他のクライアントに退出の通知
      clients.each do |c|
        c.puts "通知: #{name} が退出しました"
      end

      client.close
```

3.2. server.rb

まず、”gets.chomp”を使って、ユーザ名の取得を行う。ユーザ名の重複を確認するプログラムを表4に示す。”ERROR:”という文字が返ってきた場合、sockをcloseして終了するようにした。

表4: ユーザの重複を確認するプログラム

```
# ユーザ名の重複確認
response = sock.gets&.chomp
if response && response.start_with?("ERROR:")
  puts response
  sock.close
  exit
end
```

メッセージの送信と終了するときのプログラムを表5に示す。”gets.chomp”でメッセージを受け取り、sockにputsする。”exit”と入力されたらbreakすることで終了するようにした。

```
# メッセージ送信
loop do
  print "> "
  input = gets.chomp
  # "exit"と入力したら接続を切る
```

```
  if input == "exit"
    break
  end
  sock.puts input
end
```

4. 感想

初めてサーバプログラミングをしたのでなかなか難しかった。今回はパスワードの認証を省くために一度使用したユーザ名で再度参加することはできないようにしたが、パスワード認証について学んで実装したり、メッセージのログを残したりなどができるようになればいいなと思った。