課題 2

1270360:稗田隼也

2 月 5 日

1 仕様

プログラムは標準入力からのユーザー入力をサーバーへ送信し、同時にサーバーからのメッセージを受信して画面に表示する。サーバーとの接続にはソケットを使用し、ネットワーク通信を制御するために select() を利用している。これにより、ブロッキングせずに標準入力とサーバーの両方の入力を同時に監視し、適切に処理できるようになっている。プログラムは、まず getaddrinfo() を使用してホスト名を解決し、サーバーのアドレス情報を取得する。取得したアドレスを基に socket() を用いてソケットを作成し、connect() を呼び出してサーバーへ接続を試みる。接続が成功すると、通信の準備が完了する。標準入力からユーザーがメッセージを入力すると、それがサーバーに送信される。一方、サーバーからメッセージが送信されてきた場合は、それを受信し、画面に表示する。ユーザーが EOF（Ctrl+D）を入力した場合、あるいはサーバーが切断した場合には適切に処理を行い、プログラムを終了する。終了時には開いているソケットを適切に閉じ、リソースを解放する。このプログラムは、select() を利用することで、標準入力とソケットの両方を監視し、データが届いたときに適切に処理できるようになっている。これにより、プロセスを分岐することなく、単一のプロセス内で非同期的に通信を処理できる設計となっている。

2 説明

プログラムの冒頭では、必要なライブラリをインクルードし、ソケット通信やエラーハンドリング、シグナル処理に必要な機能を提供する準備を行う。特に signal(SIGPIPE, SIG_IGN) を設定することで、サーバーが突然切断された際に SIGPIPE シグナルが発生しないようにしている。これにより、クライアントが異常終了することを防いでいる。main 関数では、まずコマンドライン引数を解析し、ホスト名とポート番号を取得する。getaddrinfo() を使用してホスト名を IP アドレスに変換し、接続可能なアドレス情報を取得する。その後、取得したアドレス情報を基に socket() を用いてソケットを作成し、connect() を呼び出してサーバーへ接続を試みる。接続が成功すれば、通信が開始される。プログラムのメインループでは、select() を使用して標準入力（ユーザーの入力）とサーバーからのメッセージを同時に監視する。select() を用いることで、プログラムがブ

ロックされることなく、どちらの入力が発生しても即座に処理を実行できる。標準入力からデータが入力された場合、fgets() を使用してユーザーの入力を取得する。その後、改行を削除し、write_all() 関数を用いてサーバーに送信する。write_all() は、すべてのデータが確実に送信されるまで繰り返し write() を呼び出すように設計されている。途中でシグナル割り込みが発生した場合でも、安全に再試行を行う。一方、サーバーからのデータが送信されてきた場合は、read() を使用してデータを受信し、標準出力に表示する。受信データはバッファに格納され、文字列の終端を適切に処理した上で画面に表示される。もし read() が 0 を返した場合、それはサーバーが接続を閉じたことを意味するため、適切なメッセージを表示してプログラムを終了する。プログラムの終了処理として、fgets() の戻り値が NULL である場合（ユーザーが Ctrl+D を入力した場合）や write_all() の送信が失敗した場合、またはサーバーからの read() が 0 を返した場合には、ソケットを閉じ、適切な終了メッセージを表示する。このように、クライアントが予期せぬ終了をすることなく、適切にリソースを解放できるようになっている。このプログラムの特徴は、select() を活用することで、シンプルな構造ながら非同期通信を実現している点にある。通常の read() や write() を単独で使用すると、データが来るまで待機してしまうため、並行して処理を行うことが難しい。しかし select() を用いることで、標準入力とサーバーからのデータを同時に待ち受けることができ、効率的なデータの送受信を実現している。また、プロセスの分岐（fork()）を必要とせず、単一プロセスで動作するため、リソースの節約にもつながる。この設計により、クライアントはサーバーとリアルタイムで通信を行いながら、ユーザーの入力を随時処理することが可能になっている。拡張の余地もあり、例えば select() のタイムアウト機能を活用することで、一定時間操作がなかった場合に自動的に切断する機能を追加することもできる。また、サーバーとの通信内容をログに記録する機能を追加することで、通信の履歴を残すことも可能である。このプログラムは、シンプルながらも実用的な TCP クライアントの基礎を提供しており、ネットワークプログラミングを学ぶ上での良い実装例となる。

3 実用例

このプログラムを実行する前に、簡易的なサーバーを自分の PC で起動することができる。netcat（nc）コマンドを使用することで、簡単にサーバーを立ち上げることができる。例えば、以下のコマンドを実行すると、20021 番ポートで待ち受けるサーバーが起動する。

Listing 1　switch

```
1 $ nc -k -l -p 20021
```

このコマンドは、nc（netcat）を使用して、-k オプションで接続を継続的に受け付け、-l オプションで指定したポート（ここでは 20021）でリスンを行う。これにより、クライアントからの接続を待ち受ける簡易的なサーバーが動作する。次に、クライアントプログラム（上記の C 言語のプログラム）を実行して、実際にこのサーバーに接続することができる。

4　感想

　今回の課題でクライアントのプログラミングを学ぶことで通信の流れと仕組みを明確に理解できた