情報工学実験C ネットワークプログラミング

氏名:大西　隼也
学籍番号:09427510

出題日: 2017 年 12 月 12 日
提出日:2018 年 1 月 30 日
締切日:2018 年 1 月 30 日

1　概要

本実験では，基本的な通信方式である TCP/IP，UDP/IP によるネットワークプログラミングについて学習する．また，分散システムの基本的な形式であるクライアントサーバモデルの仕組みを学習する．最終的に，クライアントサーバモデルに基づくプログラムを作成する．

2　クライアント・サーバモデルの通信の仕組みについて

クライアントサーバモデルとは，クライアントとサーバを分離して管理するソフトウェアモデルであり，今回の実験ではローカルの環境下で動作するプログラムを作成したが，ここでは，代表的なサーバモデルである，メールサーバや web サーバとクライアントがインターネットを通じて通信する際の方法について概要を説明した後，実装したプログラムのソケットを用いた通信について述べる．

2.1　インターネットでの通信の仕組み

インターネット上のすべての計算機には，一意の IP アドレスが割り振られているため，IP アドレスを用いて計算機を特定することができる．しかし実際に IP アドレスを用いて通信を行おうとすると，例えば岡山大学の IP アドレス (150.46.30.130) など数字の羅列で人間が直感的に分かりにくいため，インターネット上のホスト名（www.okayama-u.ac.jp）と IP アドレスを対応させるシステム，DNS(Domain Name Service) が用いられている．

DNS もサーバの一種であり，ターミナル上で nslookup コマンドなどを用いて処理を依頼すると，ホスト名から IP アドレスに（正引き），IP アドレスからホスト名に（逆引き）の変換結果を返す．

また同時に複数の計算機と通信する際や通信相手計算機に複数のプログラムが存在する場合には，IP アドレスに加えて補助アドレスとしてポート番号を利用する．

ポート番号とは，0-65535 の間で指定可能な数であり，サービスの種別を判断するために用いられる．例えば，IP アドレス（150.46.30.130）とポート番号（80）は岡山大学の web サーバを示す．

自作でプログラムを作成する際に注意したいのは，1023 番までのポートは well-known ポートと呼ばれ，主要なプロトコルで用いられる番号が決まっているため，それ以外の番号を利用する必要が有る．

- DNS(53)，HTTP(80)，POP3(110) など

2.2　今回作成したクライアントサーバモデルでの通信

以上述べてきたように，インターネットは TCP/IP という通信プロトコルを用いて通信を行っているが，実際に TCP/IP をプログラムから利用するには，プログラムとインターネットをつなぐ出入り口が必要になってくる．その出入り口となるのがソケットと呼ばれるものであり，TCP/IP 通信はソケット通信と呼ばれることもある．

ソケットの最大の特徴として，ソケットを介してデータを送受信する際の要領が基本的にファイル入出力と同じであり，扱いやすいという利点がある．そのため単純なプロセス間通信では，通信相手プロセスとの間にソケットを生成し，そのソケット番号を通信に利用しながらソケットに対して送信や受信の命令を実行することでデータの送受信を実現している．

3　名簿管理プログラムのクライアント・サーバプログラムの作成方針

3.1　名簿管理プログラムの仕様について

基本的にはプログラミング演習で作成した名簿管理プログラムの入出力部分を send 関数や recv 関数を用いて書き換えを行い，サーバ，クライアント間で通信を行えるように実装し直す．

今回与えられた仕様として，名簿管理プログラム終了時，クライアントのみ終了し，サーバ側のプログラムは接続待機状態に戻り待つというものがあったので，通信部を while 文で意図的に無限ループするような方針でプログラムを作成した．

4　プログラム及び，その説明

4.1　TCP/IP のプロトコルの説明

IP は，Internet Protocol の略称で，データグラムを転送するためのプロトコルとされている，アドレスとして IP アドレスとポート番号を用いる．

TCP は，Transmissiomn Control Protocol の略称で，ストリーム転送サービスを提供しており，これにより，信頼性と複数回に分けて送り出したデータについても，順序を保証することが可能になっている．しかし，その分通信に時間がかかるというデメリットもある．

4.2 名簿管理プログラムのコマンド一覧

コマンド	意味	備考
%Q	終了 (Quit)	
%C	登録件数などの表示 (Check)	
%P n	先頭から n 件表示 (Print)	n=0: 全件表示, n < 0 後ろから -n 件表示
%R file	file から読み込み (Read)	
%W file	file へ書き出し (Write)	
%F word	word を検索 (Find)	結果を%P と同じ形式で表示
%S n	データを n 番目の項目で整列 (Sort)	表示はしない
%D n	データを n 件削除 (Delete)	仕様は後述する
%A n	n 番目にデータを登録 (Add)	
%B	直前の状態に戻る (Back)	%R,%A の使用後のみ
%M	各コマンドの仕様を表示 (Manual)	

4.3 クライアントの処理の流れ

ここでは，クライアントプログラムの主な処理の流れと, 今回使用した TCP/IP の関数について述べる.

1. 通信相手の IP アドレスを取得
   - gethostbyname:IP アドレスを得る
2. ソケットの作成
   - socket:ソケットを作成する
3. 接続の確立
   - connect:コネクションを確立させる
4. 要求メッセージを送信
   - send:メッセージを送信する
5. 応答メッセージを受信
   - recv:メッセージを受信する
6. 応答メッセージを処理
7. ソケットの削除
   - close:ソケットを削除する

4.4 サーバの処理の流れ

サーバは要求メッセージの到着を常に待ち，要求メッセージが到着したら処理を行い，結果を送信する．ここでは，サーバプログラムの主な処理の流れと，今回使用した TCP/IP の関数について述べる．

1. ソケットの作成
    - socket:ソケットを作成する
2. ソケットに名前をつける
    - bind:ソケットに名前をつける
3. 接続要求の受付を開始する
    - listen:接続要求を待つ
4. 接続要求を受け付ける
    - accept:接続要求を受け付ける
5. 要求メッセージを受信
    - recv:メッセージを受信する
6. 要求メッセージを処理
7. 応答メッセージを送信する
    - send:メッセージを送信する
8. 次の接続要求の受付を開始する

5 プログラムの使用法

5.1 クライアントプログラムの動作

クライアント側のプログラムは起動後，ソケットを作成し，サーバ側との通信を確立した後，コマンドの入力待ちを行う．コマンドが入力されるとソケットを介してコマンドをサーバ側に送信し，処理結果を受け取り表示を行う．

```
oonishishunya-no-MacBook-Air:network oonishishunya$ ./meibomac-client localhost
クライアントの入力待ち
%C
登録件数は 0 件です.

クライアントの入力待ち
%R sample.csv
読み込みが完了しました．%C 等で確認してください.

クライアントの入力待ち
```

%C
登録件数は 2886 件です.

クライアントの入力待ち
%Q
終了します。

5.2 サーバプログラムの動作

サーバ側のプログラムは起動後，ソケットを作成し通信の準備が整ったらクライアントからの入力を待ち，%Q 以外のコマンドを受け取った際には処理を開始し結果をクライアントに返す.
%Q コマンドを受け取った際には，処理を行わず通信待機状態へと戻る.

```
oonishishunya-no-MacBook-Air:network oonishishunya$ ./meibomac-server
クライアントの入力待ち
入力 %C
サーバの処理開始
入力 after parse_line(): %C
処理終了

クライアントの入力待ち
入力 %R sample.csv
サーバの処理開始
入力 after parse_line(): %R sample.csv
処理終了

クライアントの入力待ち
入力 %C
サーバの処理開始
入力 after parse_line(): %C
処理終了

クライアントの入力待ち
入力 %Q
処理終了
```

6 プログラムの作成過程に関する考察

6.1 工夫した点

6.1.1 ソケットの再送待機状態対策

ソケットの特性上，一度ソケットをクローズすると，最初にクローズした側（今回の場合はクライアント側）は，再送待機状態 (TIME_WAIT) になる．この状態では，プログラム動作外で到着したパケットを破

棄できるよう，数分間は CLOSED 状態にならないので，他のソケットがそにポートを使用することができない．対策法としては，最も単純なのがプログラムを起動するたびに使用するポートを変えるなどが考えられるが，今回は作成したソケットに SO_REUSEADDR オプションを付加することで問題を解決した．

SO_REUSEADDR オプションを付加することで，同じローカルアドレスに bind を行ってもエラーにならず処理を行うことができる．

```
/* SO_REUSEADDR をつける*/

int ret;
ret = setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR,(const char *)&yes, sizeof(yes));
if(ret < 0){
  printf("Error : can't opt");
  return 0;
}
```

6.2 作成に苦労した点

6.2.1 %P コマンドの実装について

%P コマンドに代表されるように，サーバ側のプログラムから複数件の結果が返される際，パケットの分割サイズを予測することができないため，%P コマンドの出力件数が一つずれたり，send と recv のデータ送受信の関係からコマンド実行後，意図しない出力が行われコアダンプを起こすなどの問題が頻発した．

そこで，%P コマンドを入力した際には，実際にプリント処理を行う，recv，send のとは別に，何回プリント処理を行うか回数についてのみのデータの送受信を行うよう実装を行い，その後そのループ回数だけ recv を行うようクライアントプログラムを記述することで，問題を解決することができた．

```
  /*メッセージを受信する*/
  char kekka[MAX_LINE_LEN + 1];
  if((line[0]=='%' && line[1]=='P') || (line[0]=='%' && line[1]=='F')){
    bzero(&kekka, sizeof(kekka));
    recv(sockfd, kekka, sizeof(kekka), 0); //回数を受け取る
    int times;
    int l;
    times = atoi(kekka);
    for(l=0; l<times; l++){
bzero(&kekka, sizeof(kekka));
recv(sockfd, kekka, sizeof(kekka), 0);
printf("%s\n", kekka);
    }
```

7　得られた結果に関する考察

7.1　recv と send の対応付けに関する問題

今回の実験は，クライアントサーバモデルを構成して以前作成したプログラムを動かすというものだったので，実際の名簿管理プログラムの速度的な性能や機能面などの考察はテーマに沿わないため，6 章の作成過程に関する考察が中心となった．ここでは，プロセス間通信で想定外の動作を防ぐため特に注意し，実装に手間取った recv と send の対応付けに関する問題について考察する．

自作の名簿管理プログラムのコマンドの中で，recv と send の対応付けが一対一対応にならないものは，%P，%F，%A，%B の 4 つであった．

%P や%F の場合，問題になるのはプリント回数のみであるので，前述の考察で実装したように，プリント回数とプリント処理部をわけて通信を行うといった対応が可能だったが，%A や%B など自作で機能拡張したコマンドなどはその関数の実装法などが一般的ではない場合もあるため，今回はクライアント側ですべて場合分けをしてコマンドごとに recv と send の回数を合わせにいったが，現実的ではないことがわかる．

インターネットやメールサーバなどユーザ数や同時アクセスの問題を考えると，個々の処理はその関数の中で完結するように実装を行い，ユーザインタフェースにあたる，クライアント部などのプログラムは単純に記述を行うことが，後々の仕様変更や保守の観点から見れば重要であると考えられる．

7.2　考察に付随した感想

自分の作成したプログラムでも 1 年前ともなると何を書いているか，関数がどう動いているかわからない部分が多く，コメントやレポートを丁寧に書いておくことの重要性を実感した．また，先ほど考察したように，理想的には，名簿管理プログラムの内部で処理を完結させるような構造にしたかったが，当時プログラムが動けばさえすればいいと思ってコードを書いていた部分も多くあり，今回のように後から変更が加えづらい点でとても苦労したので，今後は保守性も意識してプログラムを作成しようと感じた．

8　作成したプログラム

今回，作成したプログラムのソースコードについて，名簿管理の処理を行うメインのプログラムに加え，その通信部分を担うクライアント，サーバプログラムの 3 つに分かれており，非常に膨大なページ数となるため github へのリンクと，プログラム名を記載することで割愛する．

`https://github.com/Shunya-Onishi/network`

- 名簿管理プログラム本体:meibo-prog.c
- 名簿管理プログラムクライアント:meibo-client.c
- 名簿管理プログラムサーバ:meibo-server.c