情報科学演習Ｃ　課題2レポート

氏名　山久保孝亮
所属　大阪大学基礎工学部情報科学科ソフトウェア科学コース
メールアドレス　u327468b@ecs.osaka-u.ac.jp
学籍番号　09B22084

提出日　2024 年 5 月 27 日
担当教員　平井健士, 中島悠太

1 課題 2-1

1.1 アルゴリズム

今回私が作成した echoclient プログラムは以下のような流れで処理を実行する.

1. 入力形式の確認
2. ソケットの生成
3. ホストが存在するかどうかを確認
4. ソケットアドレス再利用の指定
5. ソケット接続先の情報設定
6. ソケットをサーバーに接続
7. ユーザからメッセージを取得.7$\tilde{9}$ は無限ループとする
8. メッセージをサーバーに送信
9. メッセージをサーバーから受信

以下でその詳細について述べる.

1.1.1 入力形式の確認

今回の echoclient プログラムは第一引数としてホスト名を指定するという使用法を想定している. そのため, それ以外の使用を制限するために main 関数のコマンドライン引数である argc の値を確認することで実現した.argc は指定した引数の個数+1 の値を表すのでこれが 2 出ない場合は正しい書式でないとしてエラーメッセージを表示し, プログラムを終了する.

1.1.2 ソケットの生成

ソケットを生成するには,socket システムコールを使用する.socket システムコールの引数は 3 つあり,1 つ目がドメインの種類,2 つ目がソケットの型,3 つ目が使用するプロトコルの種類を指定する. 以下のそれぞれの詳細について述べる.[2][3]

- ドメインの種類は AF_INET,AF_INET6,AF_UNIX,AF_RAW のいずれかである. これにより使用するドメイン内のアドレスの形式である, アドレスファミリーが決定される. これらのアドレスファミリーは sys/socket.h インクルードファイルによって定義されている.
- ソケットの型には SOCK_STREAM,SOCK_DGRAM,SOCK_RAW のいずれかである. これによりソケットの型が指定される.
  SOCK_DGRAM は UDP を使用したプロセスの通信を可能にする. データグラムソケットは
  SOCK_RAW は, 内部プロトコル (IP など) のインターフェースを提供し,AF_INET,AF_INET6 ドメインでサポートされている.
  SOCK_STREAM は TCP を使用したプロセスの通信を可能にする. ストリームソケットは信頼性の高い, 順序付けされた重複の無い双方向データフローを提供する.

- プロトコルは 0,IPPROTO_UDP,IPPROTO_TCP のいずれかである. ドメインの種類とソケットの型によってそれぞれデフォルトのプロトコルがあるので, たいていの場合デフォルトを表す 0 を用いればよい.

また, 正常に実行された場合は負でないソケット記述子を返し, 以上があれば-1 を返す. 今回は UDP を使用するので第一引数には AF_INET, 第二引数には SOCK_DGRAM, 第三引数には 0 を指定した. 正常に socket システムコールが終了したかどうかを確かめるために返り値を格納する int 型変数 sock の値が 0 より小さいかどうかを条件分岐で判定する. 正常に実行されていなければエラーメッセージを出力しプログラムを終了する.

1.1.3 ホストが存在するかどうかを確認

ホストが存在するかどうかを確認するために gethostbyname システムコールを使用した.[4] 引数は一つでホスト名を指す文字列である. 正常終了すると hostent 構造体へのポインタを返し, エラーが発生した場合は NULL を返す. エラーの種類は以下のとおりである.

- HOST_NOT_FOUND：引数で指定されたホストが見つからなかったとき
- TRY_AGAIN：ローカルサーバが権限サーバから応答を受信しなかったのでもう一度試すよう言われた時
- NO_RECOVERY：リカバリー不能エラー
- NO_ADDRESS：要求されたホスト名は有効だが DNS サーバで IP アドレスが見つからなかったとき
- SERVICE_UNAVAILABLE：指定されたネームサービスが実行されていないか使用可能ではないとき

また,hostent 構造体は以下のようなメンバを持っている.[5]

- h_name：ホストの名称
- h_aliases：ホストの代替名のリスト
- h_addrtype：返されるアドレスのタイプ
- h_length：アドレスの長さ
- h_addr_list：ホストのネットワークアドレスのリスト

したがって,hostent 構造体の構造体変数のポインタとして server を定義し, これに gethostbyname システムコールの返り値を格納した. また, 引数にはコマンドライン引数の argv[1] を使用した. その後 gethostbyname システムコールが正常に終了したかどうかを判定するために server に NULL ポインタが返されていないかを条件分岐によって確認する. もし返されていればエラーメッセージを出力しプログラムを終了する.

1.1.4 ソケットアドレス再利用の指定

setsockopt 関数を使ってソケット関連のオプションを設定する. 引数は 5 つあり, いかにその詳細を記述する.[6][7]

- 第一引数：オプションの適用先であるソケット
- 第二引数：オプションが設定しているレベル. 様々なレベルがあるが,SOL_SOCKET は第一引数で指定したソケットのオプションのレベルが得られる.

- 第三引数:指定するソケットオプションの名前. 様々なオプションがあるが, 今回使用する SO_REUSEADDR について記述する. これは通常一定時間ほかのソケットがそのポートを使えなくなってしまうことを防ぐことができるようになる.[8]
- 第四引数：オプションデータへのポインタ.SO_RESUSEADDR を有効にするには整数型で 1 の値が渡される.
- 第五引数：オプションデータの長さ.

また, 返り値は正常に実行された場合は 0 を返し, されなかった場合は-1 を返す. したがって, 今回作成したプログラムでは int 型変数 reuse を 1 に設定し,setsockopt 関数の引数に使用して正常に実行されたかを確認するために返り値が 0 未満かどうかで条件分岐をさせた. 正しく実行されていなかった場合はエラーメッセージを出力しプログラムを終了した.

1.1.5 ソケット接続先の情報設定

ここでは,hostnet 構造体と sockaddr_in 構造体のメンバに情報を格納している.sockaddr_in 構造体のメンバは以下のようになっている.[9]

- sin_family：アドレスファミリを指定
- sin_part：ポート番号を指定
- sin_addr：IP アドレスを in_addr 構造体で指定
- sin_zero[8]：OS の内部仕様専用

sockaddr_in 構造体の構造体変数名は svr とした. 以下のプログラムのように情報を設定した. 1 行目では svr のすべてのメンバを bzero 関数を使ってゼロに初期化している.2 行目は svr のメンバ sin_family に AF_INET を設定している.1.1.2 より, ソケットのアドレスファミリーを指定し, このソケットが IPv4 を使用すると設定する. 3 行目ではサーバーの IP アドレスを 1.1.5 で取得し,server のメンバに格納されているサーバーの IP アドレスを svr のメンバに bcopy 関数を使ってコピーする.4 行目では svr のメンバにポート番号を指定している.

1.1.6 ソケットをサーバーに接続

ここでは connect システムコールを使ってソケットをサーバに接続している.connect システムコールの引数は以下のようになる.[10]

- s：接続されていないソケットを識別する記述子
- name：接続を確立する必要がある sockaddr 構造体へのポインター
- namelen：name パラメータがさす sockaddr 構造体の長さ

また, 返り値は正常に終了した場合は 0 を, 異常が発生すると-1 を返す. 今回のプログラムでは第一引数に sock, 第二引数に svr へのポインタ, 第三引数に svr 構造体の長さを格納している. そして connect システムコールの返り値が 0 より小さければ異常が発生したとしてソケットを閉じてプログラムを終了する.

1.1.7　繰り返し処理

今回のプログラムの仕様では標準入力から読み込んだ文字列をサーバプログラムへ送信し受信した文字列を EOF を受け取るまで繰り返すというものであった．したがって, 以下の 1.1.8 と 11.9 で記述するメッセージの送信と受信は EOF が入力されるまで繰り返し実行される必要があるから,while 文を使ってそれらの処理を無限ループにした．つまり, 各ループごとにユーザに fgets 関数を使って標準入力から文字列を配列 rbuf に格納し, それをサーバに送信, 受信するというアルゴリズムを採用した．

1.1.8　メッセージをサーバーに送信

ここでは write システムコールを使ってソケットにデータを送信している.write システムコールの引数は以下のようになる.[12]

- fs：ソケット記述子
- buf：書き込まれるデータを保留するバッファを指すポインタ
- n：buf パラメータが指すバッファの長さ

返り値は, 正常に実行されたときは 0 以上の値を返し正常に実行されなかった場合は-1 を返す．fs には sock,buf には rbuf,n には strlen 関数を使って取得した buf の長さを設定した. そして変数 n に write システムコールの返り値を格納する．n が 0 より小さければ異常に終了したと判定しソケットを閉じてプログラムを終了する．

1.1.9　メッセージをサーバーから受信

ここでは read システムコールを使ってソケットからデータを読み取っている.read システムコールの引数は以下のようになる.[13]

- fs：ファイルまたはソケットの記述子
- buf：データを受け取るバッファへのポインタ
- n：buf パラメータがさすバッファの長さ

返り値は, 正常に実行されたときは読み込んだバイト数を, 異常終了した場合は-1 を返す.[14] 今回のプログラムでは fs に sock,buf に rbuf,n には strlen 関数を使って取得した buf の長さを設定した．そして変数 n に read システムコールの返り値を格納する．n が 0 より小さければ異常に終了したと判定しソケットを閉じてプログラムを終了する．

1.2　実行結果

2　課題 2-2

2.1　アルゴリズム

今回私が作成した simple-talk-server プログラムは以下のような流れで処理を実行する．

1. ソケットの生成
2. ソケットアドレスの再利用の設定

3. クライアント受け付け用ソケットの情報設定
4. ソケットアドレスの割り当て
5. 待ち受けクライアント数の設定
6. クライアント受け付け
7. クライアントホスト情報の取得
8. select システムコールの設定
9. select システムコールによって以下の処理のどちらを行うか判定する.
    - 読み込みクライアントに送信
    - ソケットから読み込み

また,simple-talk-client プログラムは以下のような流れで処理を実行する.

2.2 実行結果

3 発展課題

4 感想

今回の課題を通して UDP 及び TCP への理解が深まり, またプログラムを使ってどのようにサーバーとクライアントが接続されるのかを理解することができた. 個人的にネットワークには興味があるので, 今後の課題も興味を持ちながら取り組んでいきたいと思う.

5 謝辞

今回の課題を通して質問対応, レポート採点等をしてくださった教授,TA の皆様方ありがとうございました. 今後の課題もよろしくお願いいたします.

参考文献

[1] 情報科学演習 C 指導書

[2] https://www.ibm.com/docs/ja/zos/2.5.0?topic=functions-socket-create-socket 5/21 アクセス

[3] https://ryuichi1208.hateblo.jp/entry/2020/03/20/144644 5/21 アクセス

[4] https://www.ibm.com/docs/ja/aix/7.3?topic=g-gethostbyname-subroutine 5/21 アクセス

[5] https://www.qnx.com/developers/docs/8.0/com.qnx.doc.neutrino.lib_ref/topic/h/hostent.html 5/21 アクセス

[6] https://www.ibm.com/docs/ja/zos/2.4.0?topic=functions-setsockopt-set-options-associated-socket 5/21 アクセス

[7] https://docs.oracle.com/cd/E19455-01/806-2730/sockets-49/index.html 5/21 アクセス

[8] https://qiita.com/bamchoh/items/1dd44ba1fbef43b5284b 5/21 アクセス

[9] https://www.opto-support.com/oph5000sdk/ja/BuiltIn_WLAN_Library_Structures_sockaddr_in.html 5/27 アクセス

[10] https://learn.microsoft.com/ja-jp/windows/win32/api/winsock2/nf-winsock2-connect 5/27 アクセス

[11] https://ja.manpages.org/connect/2 5/27 アクセス

[12] https://www.ibm.com/docs/ja/zos/2.5.0?topic=functions-write-write-data-file-socket 5/27 アクセス

[13] https://www.ibm.com/docs/ja/zos/2.5.0?topic=functions-read-read-from-file-socket 5/27 アクセス

[14] https://cgengo.sakura.ne.jp/read.html 5/27 アクセス