情報工学実験C ネットワーク実験課題レポート

―クライアントサーバーモデルで動作する名簿管理プログラムの作成―

氏名: 寺岡久騎 (TERAOKA, Hisaki)
学生番号: 09B22433

出題日: 2024年12月13日
提出日: 2025年 1月21日
締切日: 2025年 1月21日

1　プログラムの処理の概要および作成方針

本実験では，ネットワーク通信により動作するクライアントサーバーモデルの TCPによる相互のデータ送受信を行う名簿管理プログラムの作成を行った．本章ではサーバ・クライアントプログラムのそれぞれの機能と，プログラム全体としての処理の概要について解説する．

1.1　プログラム全体の処理の流れと概要

プログラム全体の処理の流れとして，まずサーバはクライアントからのTCPの接続要求を待ち受け，クライアントはサーバに接続する．コネクションを確立することで名簿管理の処理が開始し，以降はシステムコール関数を用いたソケット通信により以下の一連の処理を繰り返す．

1. クライアントが名簿管理に関する標準入力からの入力をサーバにメッセージとして送信する
2. サーバがクライアントからのメッセージを受信する
3. サーバがメッセージに対応した名簿データに関する処理を行い，その結果をクライアントにメッセージとして送信する
4. クライアントがサーバーから受信したメッセージを標準出力へ出力する

クライアントとサーバは一方が送信処理をする際にもう一方が受信処理を行う様に通信を同期してやりとりを行う．処理の終了時にどちらも接続を切断し，クライアントはプログラムを終了し，サーバは次のクライアントの接続を待ち受ける状態となる．

プログラムで扱う名簿データは「ID，氏名，誕生日，住所，備考」の項目からなるCSV形式であり，サーバはこの形式のデータを受信するとこれを新たな名簿データとしてメモリに保存する．また，'%'から始まるメッセージは名簿データに対する様々な処理を実行するためのコマンド入力であり，サーバからクライアントへのメッセージは主にこのコマンド処理の結果である．以下，実装したコマンドに対応した処理や機能，及び処理結果の出力について示す．

■%Q コマンド　クライアントプログラムを終了し，サーバープログラムはクライアントとの接続を終了して次のクライアントプログラムの接続を待ち受ける状態となる．

■%C コマンド　登録した名簿データ数の表示を行う．サーバからクライアントへ登録されている名簿データの数と，登録可能な残りのデータ数に関するメッセージを送信する．

■%P コマンド　引数に入力された整数値に応じて登録されている名簿データを表示する．サーバはクライアントへ該当の名簿データ全ての各項目を表示した文字列をメッセージとして送信する．引数の仕様を以下に示す．

- 0の場合：登録順に全件表示
- 負の場合：登録データの後ろから順に絶対値の数だけ表示
- 絶対値が総データ数以上の場合：登録順に全件表示

■%R コマンド　サーバ側で引数に指定された名前のファイルを読み込み，CSV形式で記された文字列を名簿データとして登録を行う．

■%W コマンド　サーバ側で引数に指定された名前のファイルへ，登録されている全ての名簿データをCSV形式で出力する．

■%S コマンド　引数で指定された整数値に対応する項目に関して，サーバプログラムのメモリに登録されている名簿データのソートを行う．

■%F コマンド　引数に入力された文字列と完全に一致する項目を持つ名簿データ全ての表示を行う．%Pコマンドと同様に，サーバは該当の名簿データの情報をメッセージとしてクライアントへ送信する．

■%D コマンド　引数に入力された整数値に応じて，サーバがメモリに保存している名簿データを削除する機能を持つ．引数の仕様を以下に示す．

- 0の場合：全データ削除
- 正の場合：先頭から引数の数だけ削除
- 負の場合：登録データの後ろから順に引数の絶対値の数だけ表示
- 絶対値が総データ数以上の場合：全データ削除

1.2　クライアントプログラムの機能と作成方針

クライアントは始めにサーバとのTCPコネクションの確立を行う処理をする．接続にはサーバが動作する計算機のホスト名と，サーバプログラムのポート番号が必要であり，これらの情報はプログラム実行時の引数として入力し，これをそれぞれIPアドレス，ポート番号として接続の処理に利用するという設計にした．

サーバとの接続が確立した後は，標準入力からの入力された文字列をサーバへメッセージとして送信し，その後サーバの処理結果を受信して標準出力へ出力するという処理を繰り返す．この繰り返しの実現には，TCPの接続処理の後にループの構文を設け，その中でメッセージの入力と送信，受信と出力の処理を行うという方針を取った．

%Qコマンドによるプログラムの終了は，処理の複雑化を避けるため，上記のサーバとのメッセージの送受信・同期の処理から分岐せず，サーバからのメッセージを受信した後にループから抜けると

いう設計にした.

1.3 サーバプログラムの機能と作成方針

サーバープログラムは，まず通信相手となるクライアントからの接続要求を待ち受ける．この機能ではクライアント側が接続をするために，プロセスのポート番号を設定する必要があり，これをプログラム実行時の引数として入力された文字列を整数値として変換することで実装を行った.

クライアントとの接続後は以下の処理を行う.

1. クライアントの接続を受け付ける
2. クライアントからのメッセージを受信する
3. メッセージに対応した名簿管理の処理を行う
4. 処理の結果をクライアントに送信，2の処理へ
5. クライアントとの接続終了後，1に戻る

処理2–4はクライアントプログラムと同様に，クライアントとのやり取りが終了するまで繰り返されるため，ループ処理を設けてその中に実装する方針とした．クライアントから%Qコマンドが送信された場合は接続の終了を示すメッセージを送信してクライアントとの接続を終了した後，プログラムはプロセスを終了せず，次のクライアントプログラムの接続を待ち受けるため，処理1–5を無限ループの処理の中に設けることで実現した．そのため，サーバプログラムの終了はキーボードのCtrl-C入力もしくは `kill`コマンドで行う． Ctrl-C入力は本来強制的にプロセスを終了させるが，本プログラムは**??** 項に示す様に安全に終了する機能を持つ.

1.3.1 多重通信受付対応機能 (発展課題2)

今回作成したサーバプログラムは複数のクライアントのプロセスからの接続を同時に処理することが可能な多重通信受け付け機能を持つ．実現にあたっては，接続の待ち受けをする処理とコネクションを確立し名簿管理のデータ通信を行う処理を分けるという仕組みが有効であると考え，接続要求の待ち受けを主に行う元のサーバプログラムを親プロセスとして TCPコネクションの確立の後にそれぞれのクライアントに対応した通信処理を行うプロセスを複製する方針により実装を行った．処理の概要については第2.1.1節に示す.

1.3.2 サーバログ機能 (発展課題3)

社会で利用されるサーバプログラムやシステムでは情報の管理・分析やセキュリティ対策のために接続元の IPアドレス等の情報，アクセスした時刻やイベントをログとして記録することが多い．そのため，本演習で作成したプログラムにおいてもサーバ側で様々な情報をログとして記録する機能を実装した.

ログに記録する情報は接続元のIPアドレスとポート番号，実行されたコマンドとメッセージにより登録された名簿データである．ログファイルの作成はクライアントとのTCPコネクション確立に行い，メッセージの送受信のループ処理の直前でその時点での時刻と接続元のIPアドレスとポート番号をファイルへ書き込む．クライアントから受信したデータに関しては，実行可能なコマンドと登録可能な形式の名簿データのみを書き込む.

1.3.3　クライアントプログラム停止時の一時データ記録と復帰処理 (発展課題3)

サーバプログラムは，クライアントとのやりとりと通信の終了を%Qコマンドのメッセージ受信により行う．しかし，クライアントプログラムがヒューマンエラー等の理由によりCtrl-C入力で終了した場合は扱っていた名簿データが保存されないまま失われてしまう．そのため，サーバ側でクライアントがCtrl-C入力で終了したことを検知し，それを異常終了とみなして一時的にファイルへその時点で保存している名簿データを書き出し，以降接続するクライアントにその一時データを読み込む選択ができる機能を実装した．

一時的に記録したファイルがある場合は新たに接続したクライアントにメッセージを送信に，そのデータを読み込むがどうかを選択させ，選択された場合はファイルの内容を名簿データとして読み込んだ上で一時ファイルを削除する．選択されなかった場合はそのファイルを削除せず，データも読み込まない．

1.3.4　Ctrl-C入力時の安全なプログラムの終了処理 (発展課題3)

サーバプログラムはクライアントの接続要求を待ち受けるために無限ループの処理を設けており，プロセスの終了には Ctrl-C入力により強制終了させる方法がある．しかし，この場合では通信で使用している待ち受け用のソケット，クライアント用のソケットの削除，1.3.2項で示したログファイルのファイルディスクリプタの削除など，本来プロセスを終了する前に行うべき処理が行われない．そのため，サーバプログラムではCtrl-C入力時にこれらの処理を行い安全に終了する機能を実装した．この機能では，上記の処理に状態のままになることを防ぐために Ctrl-C入力時にはプロセスにキーボードからの割り込みを示すシグナルが送信されるため，機能の実装にはプログラムで`signal`システムコールを利用して，このシグナル発生時に上記の行うべき処理，加えてクライアントプログラムがサーバからの受信待ち状態のままになることを防ぐためのメッセージの送信を行いプロセスを終了する方針をとった．

2　プログラムの処理の説明

サーバ・クライアントのプロセスは第1.1節で示した様にTCPコネクションを確立した後に，相互に送信と受信の対応を同期させてメッセージのやりとりを行う．これら一連の処理の流れについて

1. サーバ・クライアントのコネクションの確立
2. サーバの安全なプロセス終了処理の設定(発展課題3)
3. クライアントの標準入力からの入力データ取得・サーバへのメッセージ送信
4. サーバのメッセージ受信
5. サーバの受信メッセージによる名簿データの処理，クライアントへの処理結果のメッセージ送信
6. クライアントのメッセージ受信と標準出力への出力

に分けて解説する．

2.1 サーバ・クライアントのコネクションの確立

2.1.1 サーバプログラム

サーバプログラムは，始めにクライアントの接続要求の受け付け処理を行う．まずソケットの待ち受けの設定を行うためにsockaddr_in構造体の変数を宣言し，これに対して

- ソケットで利用するアドレスタイプにIPv4(AF_INET)を設定する
- プログラム実行時の引数で受け取った文字列をポート番号としてatoi関数により変換した整数値をhtons関数を用いてネットワークバイトオーダに変換して設定する
- どのアドレスからの要求を受け付けるために，受け付けるIPアドレスの項目をINADDR_ANYとして htonl関数によりネットワークバイトオーダに変換して設定する．

という処理を行う．次にsocket関数にソケットで利用するアドレスタイプとして第1引数にIPv4を指定するAF_INETを，第2引数に通信形式としてTCPを指定するSOCK_STREAMを設定して，ソケットの生成及びその戻り値としてディスクリプタの値を受け取る．このディスクリプタの値と先述のsockaddr_in 構造体の変数を用いてbind関数によりソケットに待ち受けるための設定を行う．

待ち受け用のソケットを作成した後，そのディスクリプタを引数として listen関数によりサーバとしてクライアントの接続待ち受けを開始する．第2引数に設定する一度に通信する上限のクライアント数は5と設定した．次に，クライアントとの接続要求の受付を行うためのループ処理に進む．この中の処理の始めで待ち受け用のソケットのディスクリプタを引数としてaccept関数で接続要求を受け取り，クライアントとの通信に使用するソケットを生成する．この時，accept関数の引数にadddinfo_in 構造体の変数のポインタをsockaddr構造体のポインタにキャストして渡し，クライアントの情報を取得し，この変数を用いて第1.3.2項で示したログ機能により open関数によるクライアント用のログファイルの作成とクライアントのIPアドレスとポート番号，時刻の記入を行う．

クライアントとの通信を行うソケットの生成を行った後，第1.3.1項で示した多重通信受付のためにfork関数によりその時点での処理の進行，メモリ状態が同じであるプロセスの複製を行う．この時の戻り値として受け取ったプロセスIDが0より大きい親プロセスとプロセスIDが0である複製された子プロセスで処理をif文により分岐させ，それぞれ以下の処理を行う．

親プロセス　接続したクライアントとのメッセージ送受信の処理は行わず，再びループの始めに戻り，accept関数で次のクライアントの接続待ち受ける．

子プロセス　始めに第1.3.3項で示した一時記録された名簿データの読み込みを選択する処理を行う．その後，クライアントとのメッセージのやりとりを行うループの処理に入る．クライアントとの通信の終了後はクライアントの接続要求を受け付けるループ処理を抜け，クライアントとの通信用ソケットをclose関数で削除してプロセスを終了する．また，この時ログ機能により作成したファイルのディスクリプタもclose関数により削除する．

これによって

- 親プロセスはクライアントの接続要求を受け続ける処理を繰り返す
- 子プロセスはクライアントとのメッセージの送受信を行い，通信終了後にプロセスを終了する

という処理の流れになり，複数のクライアントのプロセスからの接続要求を同時に処理することがで

きる．

2.1.2　クライアントプログラム

クライアントプログラムは始めに，接続先のサーバプロセスとのTCPコネクションの確立の処理をする．サーバに接続するために必要なサーバプロセスのホスト名とポート番号はプログラム実行時に引数として文字列データとして受け取るため，こを引数として getaddrinfo関数によって，ソケットの生成に必要な項目の値が設定されたaddrinfo 構造体のデータを取得する．このデータを利用してsocket関数によりサーバとの接続に利用するソケットのディスクリプタを得て，この値を引数にconnect関数によりサーバとの接続を確立する．コネクションの確立後はwhile文の中で以降解説するサーバとのやりとりを繰り返し行う．

2.2　サーバの安全なプロセス終了処理の設定(発展課題3)

サーバでは，第1.3.4項に示したCtrl-C入力時の安全なプロセスの終了処理を待ち受け用ソケットを作成した後に設けている．プログラムではsignal関数により，Ctrl-C入力時にプロセスに送信されるSIGINT シグナルを第1引数として，第2引数にプロセス終了前に行う処理をまとめた関数を渡す．この関数では，待ち受け用のソケットのディスクリプタとログファイルのディスクリプタの削除と，クライアントがサーバからのメッセージ受信待ち状態のままになることを防ぐための目印となるメッセージの送信を行う．このメッセージは1バイトで，名簿管理に関するメッセージで使用されないASCIIコードの整数値127(0x7f) を代入している．

2.3　クライアントの標準入力からのデータ取得・サーバへのメッセージ送信

標準入力からの入力データの受け取りとサーバへのメッセージの送信は同じwhile文中で行い，以下の流れで一連の処理を行う．

まず標準入力からの入力データをシステムコール関数のreadにより取得し，サイズが10バイトのバッファー(char型の配列) に格納する．そしてこのバッファーとread関数の戻り値である取得したバイト数を引数として， send関数によりサーバへメッセージを送信する．ユーザからのメッセージ入力の終了はEnterキーの押下による改行文字を目印とし， read関数の処理後に取得バイト数を利用して入力文字列の末尾の改行文字('\n')があるかを確認し，あればその要素を送信終了を示す目印としてメッセージで利用しないASCIIコードの整数値3(0x03)に変更し， send関数により送信した後にループ処理から抜ける．その後，サーバからのメッセージを受信する処理へ移行する．

2.4　サーバのメッセージ受信

クライアントからのデータ受信はクライアントの送信処理と同様にwhile文中でシステムコール関数のreadにより取得し，サイズが10バイトの受信用バッファー(char型の配列) に格納することを繰り返す．ここで，名簿管理に使用するメッセージは連結した1つの文字列データとして扱うため，予め宣言した最大入力文字数(1024)+1を格納できるchar型のメッセージ用バッファーへ読み込んだバッファーの内容をコピーを行う．この処理は以下の流れにより行う．

1. while文の直前で名簿管理処理用のメッセージ用バッファーの先頭アドレスをポインタ変数に格納する

2. while文の繰り返し処理において，read関数の処理後，戻り値の取得バイト数だけstrncpy関数により受信用バッファーの内容を先述のポインタの位置へコピーする
3. 上記ポインタ変数を取得バイト数だけ加算する．

これにより，繰り返し受信したデータを連結したメッセージとして保存を行う．受信処理の終了は，read関数の処理後に受信用バッファー内にクライアントプログラムにより付与された送信終了の目印である整数値3(0x03)があるかを確認し，ある場合はその要素を終端文字('\0')へ変更し，メッセージ用バッファーへのコピー後に受信のループ処理を抜ける．このメッセージ中の送信終了の目印により，クライアントのsendによる送信回数に依らず，サーバはメッセージの受信を適切に終了を行う．

2.5 サーバの受信メッセージによる名簿データの処理・クライアントへのメッセージ送信

メッセージの受信後，メッセージ用バッファーに格納した文字列を解析して名簿管理に関する処理を行う．この時，メッセージが実行可能なコマンドまたは登録可能な名簿データである場合は第1.3.2項で示したログファイルに書き込む．処理の結果，クライアント側にメッセージを送る場合は，クライアントの送信処理と同様にサイズが10バイトの送信用バッファーを用いて，終端文字も含めたメッセージのバイト数を10バイトずつ分割して必要な回数だけsend関数によりクライアントへデータの送信を繰り返す．

この処理の後，受信メッセージが%Qコマンドでない場合はサーバのメッセージ送信終了の目印として，整数値3(0x03)を代入した 1バイトのデータをsend関数により送信し，次のクライアントプログラムからのメッセージを受信する処理へ移行する．%Qコマンドであった場合は，このデータをクライアントへ接続終了を示す目印として，メッセージ終了の目印と同様に通常のメッセージで使用されないASCIIコードの整数値4(0x04)を代入して送信し，クライアントとのメッセージのやりとりを行うループ処理を抜け，クライアントとの通信用ソケットをclose関数により削除する．

以上の様に，コマンドにより名簿管理に関するメッセージの送信をするかどうかに関わらず，必ずクライアント側へサーバの処理が終了したことを表す目印をメッセージと送信を行う．これによりクライアントの受信処理を終了させ，送受信の同期を行う．

2.6 クライアントのサーバからのメッセージ受信・標準出力への出力

サーバからのメッセージの受信と標準出力への出力は第2.3項で解説した処理と同様に，while文中で行い，以下の流れで処理を行う．

まずサーバからの送信されたデータをrecv関数によりサイズが10バイトの受信用バッファーに格納し，この受信用バッファーのメッセージをシステムコール関数のwrite関数により，標準出力へrecv関数の戻り値である取得したバイト数だけ出力する．メッセージ出力処理の後はサーバのクライアントメッセージの受信処理と同様にバッファー内にサーバが施したメッセージの終了に関する目印に該当する要素があるかを確認し，あれば以下の処理を行う．

整数値3(0x03) メッセージの送信終了を示し，受信処理のループを抜け，再びユーザからの入力とサーバへのメッセージ送信を行う処理へ移行する．

整数値4(0x04) %Qコマンド入力判定による通信の接続終了を示し，サーバとのメッセージ送受信

処理のループを抜け，ソケットを`close`関数により削除してプロセスを終了する．

また，`recv`関数による受信データの取得後は，メッセージ出力前に第2.2節で解説したサーバ側のプロセス終了を示す目印としてメッセージに整数値127(`0x7f`) が含まれているかどうかを確認し，あれば上記の%Qコマンド入力判定時の接続終了と同じ処理を行う．

3　プログラムの使用方法と使用例

3.1　他学生のサーバプログラムとの通信

3.2　多重通信受付機能の使用

4　プログラムの作成過程に関する考察

5　得られた結果に関する考察