ソケット通信プログラミング

谷口 義樹

1　前回からの課題

- 現在の fork を用いた自作プログラムを pthread を用いたものに改良する．
- pthread を用いたチャットを完成させる．
- gdb,strace,getopt 関数のマニュアルを作成する．

2　進捗状況

2.1　fork から pthread へ

前回は，クライアントがある文字列をサーバーに投げると，サーバーが文字列を全て大文字にして，クライアントに返すというプログラムを作成し，ソケット通信についての理解を深めた．また，複数のクライアントからの接続に対応できるように，fork を用いて実装した．今回は，そのプログラムを pthread を用いたものに改良した．改良した理由は，スレッドを用いることで同じメモリスペースを共有することができるために，それぞれのスレッドからグローバル変数の読み書きが容易にでき，また動作も fork を用いたものよりも軽いことが挙げられる．

2.2　チャットの作成

作成したプログラムにさらに改良を加えて，簡易チャットを作成した．その動作を Fig. 1 に示す．

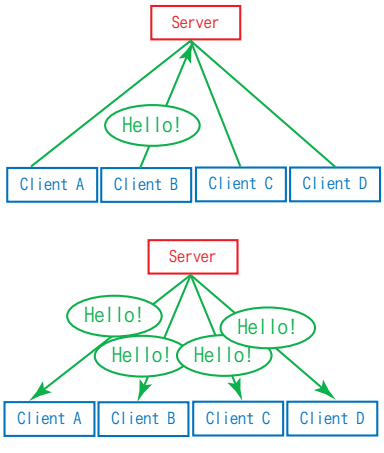

Fig. 1　chat の動作

図に示されるように，チャットを実現するためには，あるクライアントがメッセージをサーバーに送信すると，接続している他のクライアント全てに，そのメッセージが送信されなければならない．かつ，いつでもメッセージを受けつける状態にしておく必要がある．

1 つのクライアントからの接続に対し，2 つのスレッドを生成することによりこの動作を実装した．その様子を Fig. 2 に示す．複数のスレッドから同時に同じ変数にアクセスすると，データが取り出せなかったり，破壊されてしまう可能性がある．これを防ぐために，lock や unlock といった作業が必要になる．ロック状態においた場合は，それを行ったスレッドがアンロックするまで，実行を停止させることができる．

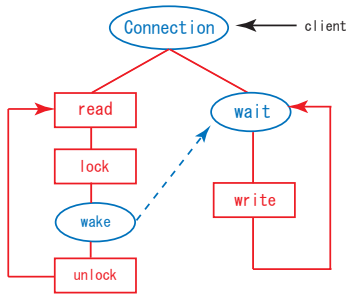

Fig. 2　2 つの thread を用いた実装

また，getopt_long() 関数を用いて，実行時のオプションで，利用するポート番号が指定できるようにした．さらに，perror() 関数を用いて，システムエラーメッセージを出力するようなエラー処理を加えた．

2.3　マニュアルの作成

GNU 開発環境における代表的デバッガである gdb(The GNU Debugger)，プロセスが呼んだシステムコールと発生したシグナルを追跡するツールである strace，コマンドラインオプション解析を行う getopt 関数に関して調査を行い，簡易マニュアルを作成，技術報告を行った．

3　翌月への課題

- DNAS について理解
- dsh の調査
- forte クラスタの再構築
- iSIGHT プロジェクト
- 用語集の作成