囲碁プログラムへの通信対局機能の組み込み方 (Unix編)

佐々木宣介

1 はじめに

この文書は Unix 系環境[1]で大会などで利用可能な通信対局機能を持つプログラムを簡単に実現する方法を解説し、囲碁プログラムを開発する際に、開発者ができるだけ思考ルーチンの開発に専念可能になる事を目指しています。対象となる読者として想定しているのは、Unix 系 OS における基本的な操作がわかり、ソースファイルからソフトウェアのコンパイル及びインストール作業を行った経験がある方といったところです。

本稿で紹介する方法を使えば、自分で作成した思考ルーチンと組み合わせて、Unix 環境上で動作する最低限の通信対局機能を持つ囲碁プログラムを作製することが可能となります。

2 必要な物

本稿で紹介するソフトウェアは以下のソフトウェアです。

- CGoban(バージョン 1.9.11)
- goDummy(バージョン 1.0.4)

CGoban および goDummy はどちらも以下のサイトで配布されています。

`http://www.igoweb.org/~wms/comp/cgoban/index.html`

今回動作確認したのは以下の環境です。

- PC-Unix 環境
  - Plamo Linux 2.0(コンパイラ：egcs 2.91.66)
  - Plamo Linux 2.2.1(コンパイラ：gcc 2.95.3)
- Unix 環境
  - Solaris2.5.1(機種：Ultra1[サン・マイクロシステムズ]、コンパイラ：gcc 2.95.2)

Plamo Linux は Slackware 系の PC-Unix ディストリビューションですが、RedHat 系など、その他の系統のディストリビューションや、BSD 系の PC-Unix でも問題なく動作すると思います。(未確認です)

今回利用した Unix マシンでは、Sun 純正の C コンパイラの環境が十分整っていなかったので、gcc を利用しました。Sun 純正のコンパイラ環境で今回利用したプログラムがコンパイル可能であるかは不明です。

2.1 CGoban

CGoban は Unix の GUI 環境である、X ウィンドウシステム環境で動作するソフトウェアです。

CGoban は SGF 形式の記譜編集ソフトウェアですが、その他に通信対局機能が組み込まれています。この通信対局の相手としては、シリアル回線の向こう側の相手だけでなく、STDIN/STDOUT 経由でそのマシンにインス

[1]「Unix 系」という表現は厳密な用語としてはどの範囲を指すのか曖昧であり、適当ではないのかもしれませんが、本稿では Unix、Linux、FreeBSD などの総称という程度の意味と考えて下さい。

トールされている GNU Go などのプログラムを直接指定して通信することも可能です。通信対局の設定として、片方のプレイヤにそのマシンにインストールされている GNU Go、もう片方のプレイヤとしてシリアル回線を選択すれば、そのマシン上の GNU Go と、シリアル回線の先のコンピュータとの対局となります。

Unix 上で動作する GNU Go は標準の状態では図 1 のようにキャラクターベースで動作し、手の入力も座標をキーボードから直接入力しなければなりませんが、この CGoban を利用する事により、GNU Go が GUI 環境のプログラムであるような使い方で対局することも可能になります。

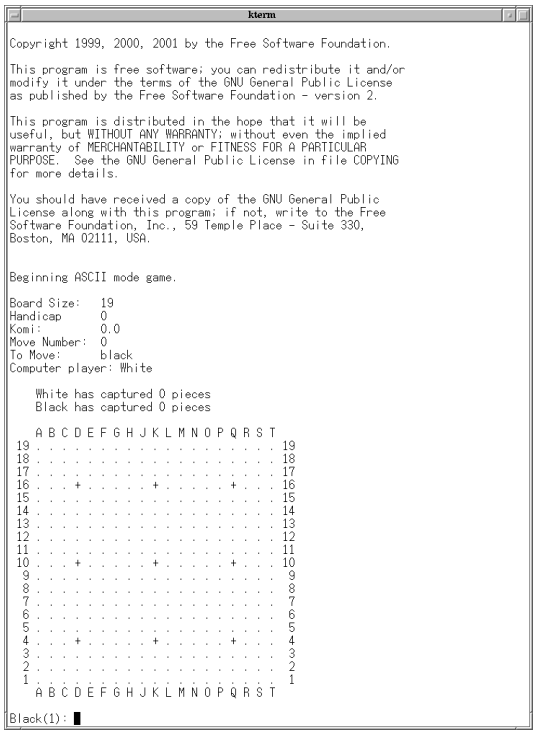

図 1: キャラクターベースで動作する GNU Go

図 2 が CGoban を利用して GUI 環境プログラムとして利用する時の動作概念図になります。

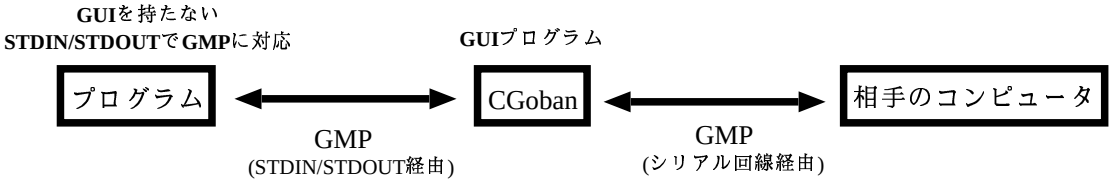

図 2: 動作概念図

CGoban の現在のバージョン 2 系列は、棋聖堂碁サーバのクライアントになっていますが、バージョン 1 系列は GPL(GNU GENERAL PUBLIC LICENSE) でソースごと公開されています。本稿ではこのバージョン 1 系列を使います。

CGoban の通信対局機能は Undo が出来ないなどの制約もあり、完全には SGMP(Standard Go Modem Protocol) を満たしてはいないようですが、恐らく大会でよく用いられる簡易版の SGMP は満たしているのではないかと思います。

CGoban は以下のような Unix では標準的な手順でコンパイル、インストールが可能です。

```
> ./configure
> make
> make install
```

2.2 goDummy

doDummy は、上記の CGoban のページで GMP(Go Modem Protocol) による通信機能を実装するためのサンプルプログラムとして公開されているものです。

このプログラムは単にランダムに座標を選択してその座標の四方に石がなければその手を選択するというアルゴリズムで次の一手を選択しています。石の捕獲を考慮していないなど、囲碁プログラムとしては、「全合法手からランダムに次の手を選択」という仕様すら満たしていません。あくまで通信機能を実装するためのサンプルとして提供されているプログラムです。

goDummy をコンパイルするためには 1ヶ所修正が必要です。ソースファイルの 1 つである gmp.c の 217 行目に問題があります。

```
sprintf(errOut, "System error %d.", errno);
```

この行で使われている変数 errno を定義しているヘッダーファイルが読み込まれていないのでコンパイルエラーになります。gmp.c のヘッダーに

```
#include <errno.h>
```

を書き加えて、必要なヘッダーファイルを読み込むように変更すればコンパイルできます。[2]

この修正を行った後、以下のように make コマンドを実行してコンパイルします。goDummy という名前の実行ファイルが生成されます。

```
> make
```

配布されている goDummy の Makefile にはインストールに関する記述がありませんので、作成した実行ファイルは、必要ならば自分で/usr/local/bin/などにコピーしてください。

2.3 CGoban と goDummy を利用した通信対局

2.3.1 cgoban の起動

まず、CGoban を起動します。コマンド名は「cgoban」です。図 3 のようなウィンドウが開きます。「New Game」及び「Load Game」は、対局をするためのメニューです。また、「Edit SGF File」というメニューが SGF 形式の棋譜を編集するためのメニューです。その他、対局サーバに接続するメニューボタンがあります。

「Go Modem」というメニューボタンが本稿で利用する通信対局用のボタンです。

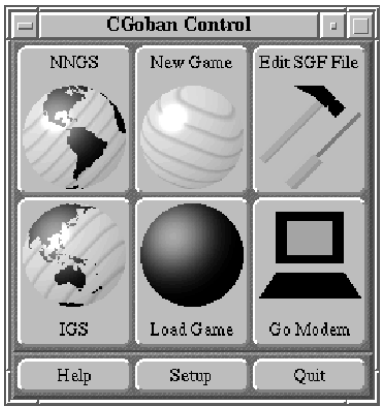

図 3: CGoban の起動画面

[2]この修正方法は CGF 会員の轟さんに教えていただきました。ありがとうございました。

2.3.2 通信対局の選択

続いて、「Go Modem」ボタンを押して通信対局モードにします。図 4 のようなウィンドウが表示されます。通信対局の設定画面です。

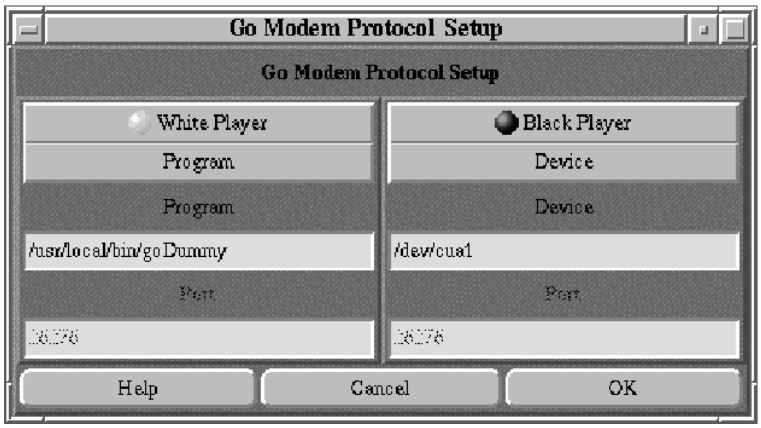

図 4: 対局モードの設定画面

この対局メニューで、片方をシリアルポート、もう一方を GNU Go(プログラム名は「gnugo」) や goDummy などのプログラムを指定してください。シリアルポートの向こうの相手と、呼び出したプログラムとの通信対戦ができます。対局のセットアップ画面で片方は「device」を指定してシリアル回線のポートを指定、もう一方は「Program」を選択して囲碁プログラムを指定すれば (例えば goDummy が/usr/local/bin/に置かれている時には「/usr/local/bin/goDummy」と指定する)、片方がシリアル回線の向こう側のコンピュータ、もう一方が指定したプログラムという通信対局を行なうことが出来ます。

シリアルポートの指定は linux では/dev/cua0 または/dev/cua1(それぞれ COM1、COM2 に対応) です。Solaris2.5.1 では、/dev/ttya または dev/ttyb(それぞれ COM1、COM2 に対応) を指定するようになります。また、システムの設定によってはシリアルポートを利用するためには管理者権限 (root) が必要となる場合がありますので、そのような場合には、管理者ユーザが CGoban を起動してください。

図 4 は、黒が通信回線 (/dev/cua1 は、Linux 環境で COM2 をあらわす)、白にプログラム (goDummy) を指定した例です。もちろん、goDummy でなく、GNU Go(プログラム名は gnugo) を指定した場合でも問題ありません。

通信対局の設定を指定して、「OK」を選択すると、図 5 のように対局の条件を設定するウィンドウが表示されます。ここでルールと時間を設定します。

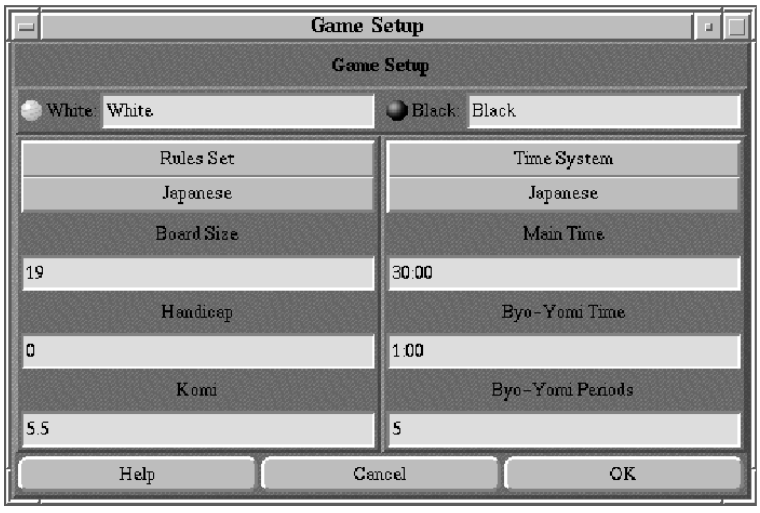

図 5: 対局条件の設定画面

「OK」ボタンを押すと、図 6 のように通信対局が開始されます。

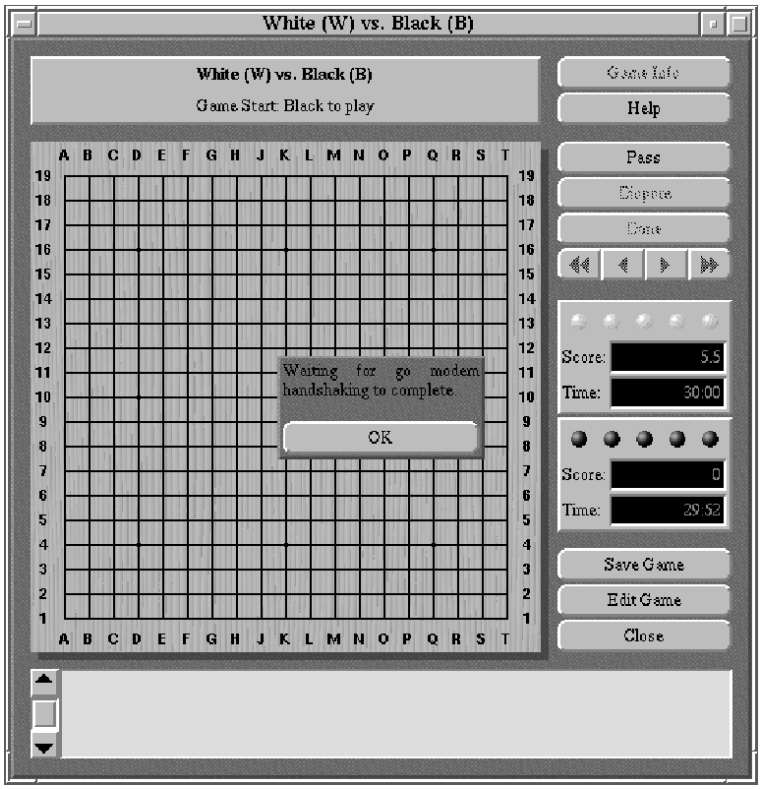

図 6: 対局開始の画面

3　自分の思考ルーチンの組み込み

では、実際に自分が作った思考ルーチンと組み合わせて自分のプログラムを通信対局対応にするためにはどのようにしたら良いのか。goDummy の次の一手生成ルーチンを自分の作成した思考ルーチンで置き換えることにより、CGoban と組合せて、自分の作った思考ルーチンを通信対局機能を持ったプログラムのように利用する事が出来るようになります。

goDummy のソースファイルはプログラムが記述されている dummy.c と gmp.c という 2 つのファイルおよびヘッダーファイルの gmp.h、dummy.h から成ります。dummy.c の中にある `playGame` という関数の中に手を決定するルーチンが含まれています。この部分を自分で作った手決定のルーチンと置き換えます。

`playGame` 関数の中で、生成した次の一手を送信しているのは以下の部分です。(`playGame` 関数の 22 行目)

```
gmp_sendMove(ge, x, y);
```

変数 x 及び、変数 y が次の一手の座標です。また、この行の下の方に相手の手を受け取るコードがあります。(`playGame` 関数の 39 行目)

```
message = gmp_check(ge, 1, &x, &y, &error);
```

この部分でも同様に変数 x 及び、変数 y が相手の手の座標です。

自分のプログラムで生成した次の一手の座標を `gmp_sendMove` で送信し、相手の手を `gmp_check` から受け取って自分の思考ルーチンに渡すように `playGame` 関数を書き換えれば自分の思考ルーチンを通信対局対応にする事が出来ます。

この他にもパスの送信をする時のサンプル (`gmp_sendPass` 関数を使う) も `playGame` 関数内にあります。以下のようにすればパスを送信します。

gmp_sendPass(ge);

簡単なプログラムのイメージは以下の図 7 のような感じになるでしょう。双方がパスをした時の対局停止処理、エラー処理などは省略しています。このようなコードを playGame 関数に組み込むことにより、CGoban と組み合わせることが可能となります。

```
while(1){
  if(自分の手番){
    次の一手生成;
    盤面情報の更新処理;
    if(パスの時){
      gmp_sendPass(ge);
    }
    else{
      gmp_sendMove(ge, x, y);
    }
  continue;
  }
  else{
    message = gmp_check(ge, 1, &x, &y, &error);
    相手の手に応じた盤面情報の更新処理;
  }
  continue;
}
```

図 7: 通信機能実装のサンプル

4 最後に

本稿は筆者の能力不足のため、まだまだ不十分な部分があるかと思いますが、この文書が、新たに囲碁プログラムを作ろうと考えている人達が、気軽に開発を始める助けとなることを願います。

間違い、改善点などがありましたら、sasaki@bus.hiroshima-pu.ac.jp までお知らせいただければ助かります。