System Architecture

* 全域變數

struct node {

char eatenchessx[250],eatenchessy[250],chess[9][9];

    struct node \*next,\*prev;

};

typedef struct node NODE;

NODE \*first,\*current,\*previous;

Char bx;

int by, ax, ay;

int numx=0, numy=0;

int count=1,player;

FILE \*cfptr;

ev\_io stop\_watcher;

ev\_timer timeout\_watcher;

time\_t xtimestart,xtime,ytimestart,ytime,xusedtime,yusedtime;

int xhour, xminute, xsecond,yhour, yminute, ysecond;

* void timer( )

包括timer、timerm(chessmove)、timerh1(hitin1)、timerh2(hitin2)、timerp(chesspromotion)。初始化ev\_io(運用stop\_watcher監控。當有io事件發生時呼叫stop\_cb)、ev\_timer(運用timeout\_watcher監控。每秒呼叫一次timeoutcb，不同的timer呼叫對應的cb)，並用xtimestart、ytimestart定現在時間。

* void timeout\_cb (EV\_P\_ ev\_timer \*w, int revents)

包括timeout\_cb、timeout\_cbm、timeout\_cbp等，對應不同timer的呼叫。用xtime、ytime定現在時間，將其與xtimestart、ytimestart相減算出經過的時間並加上此次呼叫timer之前存取的時間(xusedtime、yusedtime)，換算為幾時、幾分、幾秒，以xhour, xminute, xsecond,yhour, yminute, ysecond分別存取並輸出時間與遊戲介面。不同的timeout\_cb輸出不同的遊戲介面。

* void stop\_cb (EV\_P\_ ev\_io \*w, int revents)

當有io事件發生時會呼叫此函式。將ev\_timer及ev\_io停止並運用xusedtime、yusedtime存取累計的時間。

* void chessinitial( )

在main中會先跟電腦要一個NODE空間，並將current指向此NODE。

NODE當中的eatenchessx會儲存player X移動時目標的資料；eatenxhessy儲存player Y的；chess儲存棋盤上的棋子。一個NODE 儲存玩家下的一步棋盤與X、Y吃掉的棋子。將eatenchess的值初始化為b(表示空的)，並將每個棋子的起始位置儲存在陣列chess[9][9]的對應座標中。

* void chessprint( )

輸出current指到的NODE當中的資料，顯示出棋盤與棋子。

* void chessback( )

此為悔棋，當main中讀到bx的輸入值為0時，輸出上一個NODE (previous指到的)當中的資料，先用temp儲current的資料再將current指到previous指的NODE，previous再指到上一個NODE，最後移除temp指的空間。

* int movecheck( )

判斷輸入的座標bx,by(beforex,beforey)上是否有棋子以及判定棋子是否可移動到輸入的座標ax,ay(afterx,aftery)，若可以回傳1；不行回傳0。

* int eatchess( )

若movecheck回傳1，則進入此函式。先新增一個NODE，將current指向此空間。之後判斷目標座標上的資料是否為己方棋子，若是則回傳0；若不是則用current指向的eatenchessx(當玩家X下棋)、eatenchessy(當玩家Y下的棋)儲存目標座標上儲存的資料，再分別將numx、numy值加1，下一步棋將儲存於陣列中的下一個位置。最後回傳1，在中使count+1，改變Player。

* void chessmove( )

若movecheck與eatchess回傳值皆為1，則進入此函式。先將新NODE儲存的資料設為與上一個NODE一樣，再將新的目標座標儲存的資料(current->chess[ax][ay])改為舊的選取座標(previous->chess[bx][by])的棋子，並將新的選取座標(current->chess[bx],[by])的資料改為b(表示空的)，最後輸出此NODE的資料。

* int win( )

偵測玩家是否將對方的王或玉吃掉，若有表示該玩家勝利遊戲結束，回傳1結束遊戲，若無則回傳0。

* int chesspromote( )

當玩家成功下完一步棋後，判斷棋子是否可升變。若不行則回傳0；若可以則詢問玩家是否要升變，當讀取到turn為y時，則升變棋子並回傳1重新輸出棋盤；若turn為n時回傳0。當電腦讀取輸入時，因為有io事件發生timer會停止。所以讀取完值後再叫timer繼續計時。

* int hitin( )

若main讀到bx的輸入為H時，進入此函式，新增一個NODE(用newnode指向它，將其與上個NODE儲存相同的資料)並呼叫timerh1繼續計時。接著讀取玩家輸入值ordinal存取玩家想打入的棋子為第幾個吃掉的棋子，再呼叫timerh2繼續計時。然後讓tempeatchenchess存取eatenchess當中的棋子(去除掉b)，找出玩家選取的棋子，並用temp記錄此棋子在eatenchess當中的第幾個位置，再讀取玩家想要打入的位置(ax,ay)，最後判定打入的位置是否符合將棋打入規則。若符合則將此棋存於新NODE的chess[ax][ay]當中，並將此棋在eatenchess中的位置(eatenchess[temp])改為b，再回傳1在main中使count+1，改變player若不符合規則回傳0。

* void datawrite( )

當main中讀到bx的輸入值為S時，執行此函式。將各個NODE儲存的資料依序存於檔案chessgame.txt中，並在檔案結尾處存入0(表示讀取時NODE只存到此)，再存入雙方玩家分別遊玩的時間xusedtime、yusedtime。

* void readdata( )

當玩家的指令為讀取舊檔(./bin/shogi -l chessgame.txt)時，讀取檔案chessgame.txt存取的資料。在讀到0之前，不斷創建新NODE存取每一步的棋盤與吃掉的棋子，之後讀取遊玩時間分別存入xusedtime與yusedtime。