五子棋通用引擎协议

一、概述

五子棋机器博弈竞赛使用的竞赛平台客户端（`manager`） 遵循通用的通信协议，所有参赛的五子棋引擎（`brain`）必须遵循通用协议。
另外也可以基于通用协议开发客户端，以实现其他不同的功能。
严格遵循通用协议开发五子棋应用，可以实现可视化界面与智能引擎完全分离，使开发者专注于开发后台引擎或者可视化界面，提高软件设计效率。
本次五子棋机器博弈引擎程序开发使用该通用引擎协议，使得所开发的引擎程序可被任何遵循该通用协议的客户端加载从而方便进行人机及机机对战比赛和测试。

二、五子棋通用引擎协议的通信方式

- 该协议 使用管道（pipe）进行引擎与客户端间的通信。
- 当客户端载入引擎时，先通过 CreatePipe()函数创建两条管道，一条是输出管道(pipeFromAI)，另一条是输入管道(pipeInAI).。然后通过 CreateProcess()函数为引擎新开一个进程，并重定向到两条管道。
- 引擎使用标准输入输出 *（C 语言使用 scanf 和 printf 函数；Pascal 语言使用 readln 和 writeln 函数）*。引擎必须是一个控制台项目,而不能是一个基于可视化界面开发的项目。

注意：如果使用 C 的 printf，之后记得要调用 fflush，否则可以使用 ReadFile 和 WriteFile

- 引擎和客户端之间的每一条指令是一行（只有一个例外（BOARD 指令）），每行结尾以 CR LF（0x0d,0x0a）结束。引擎可以发送以 CR LF 或者只有 LF 或 CR 结尾的命令，客户端会进行识别，客户端会忽略空行。

三、五子棋通用引擎协议的指令解析

引擎从客户端接收的指令有：`START , BEGIN , INFO , BOARD , TURN , END` 。

其他指令使用 Unknown 回复，（这应该是为了以后可以扩展指令）

- **引擎名称要求：**

- A-Z,a-z,0-9,破折号（-），下划线（_），点（.）
- 名称必须以“pbrain-”为开头，否则引擎将通过旧协议使用文件进行通信

示例:`pbrain-swine.exe`

- 这里只要求可执行文件必须以“pbrain-”为前缀，如果打包成 Zip 则无此要求

1、重要指令

- **START** *[size]*

接收到这个指令，引擎会初始化棋盘大小和初始化一个空棋盘。棋盘大小必须能被设置成 20，因为在 Gomocup 中默认为 20（但这不是强制要求）。

- 如果无法实现 size 的棋盘则回复 ERROR，同时可以在 ERROR 后面返回信息。
- 如果成功则是回复 OK

示例:

```
The manager sends:

START 20

The brain answers:

OK

或者 ERROR message - unsupported size or other error
```

- **TURN** *[X],[Y]*

参数是引擎对手棋子的坐标。
回复是引擎思考之后的下子坐标，形式是 x，y
示例:

```
The manager sends:

 TURN 10,10

The brain answers:

 11,10
```

- **BEGIN**

客户端向先手的 Player 发送该指令。
回复是引擎思考后的下子坐标。

```
The manager sends:

BEGIN

The brain answers:

10,10
```

- **BOARD**

该指令用于继续已有棋局或者悔棋、重做等相关命令。
该命令一般在 START、RESTART 或 RECTSTART 指令之后发出来初始化一个空棋盘。

该命令有多行。命令 BOARD 后每一行格式：`[X][Y],[field]`

其中[X] 和[Y] 是坐标，[field]取 1 表示 own stone，取 2 表示 opponent's stone，或者取 3 表示 winning line 。
命令以 DONE 结束。 该命令回复一般和 TURN 或 BEGIN 命令一致。
示例：

```
The manager sends:

BOARD

10,10,1

10,11,2

11,11,1

9,10,2

DONE

The brain answers:

9,9
```

- **INFO** *[key][value]*

该指令是客户端用来发送一些信息给引擎，如果引擎并无处理此信息可忽略，但如果引擎未处理相关限制可能会因此输棋。同时引擎应该有处理一些引擎必备但客户端并未发送告知的信息的能力，如相关变量的默认值。
INFO 发送的一些关键信息：

- timeout_turn - time limit for each move (milliseconds, 0=play as fast as possible)
- timeout_match - time limit of a whole match (milliseconds, 0=no limit)
- max_memory - memory limit (bytes, 0=no limit)

- time_left - remaining time limit of a whole match (milliseconds)
- game_type - 0=opponent is human, 1=opponent is brain, 2=tournament, 3=network tournament
- rule - 0=five or more stones in a row win, 1=exactly five in a row win, 2=continuous game, 3=continuous and exactly five
- evaluate - coordinates X,Y representing current position of the mouse cursor
- folder - folder for persistent files

这些设定一般是在游戏一开始就设定,主要时间和内存限制的信息在游戏第一步之前就被发送处理了。剩余时间（time_left）是每一步都会发送的，一般在 TURN、BEGIN、和 BOARD 指令之前。
引擎端不需要回复。
示例:

```
INFO timeout_match 300000

INFO timeout_turn 10000

INFO max_memory 83886080
```

- **END**

当引擎接收到指令，引擎必须尽快终止，客户端会一直等待。如果终止耗时过长客户端则会主动终止引擎，此时引擎不应该输出任何信息给客户端，但客户端与引擎之间的通信管道直到引擎关闭才会关闭。
命令结束后引擎会删除临时文件

- **ABOUT**

引擎接收到此指令后会返回引擎的相关信息，返回信息以关键子形式组织，并以逗号隔开，关键字有 `name,version,author,country,www,email`

示例:

```
The manager sends:

 ABOUT

The brain answers:

 name="SomeBrain", version="1.0", author="Nymand", country="USA"
```

2、可选指令

- **RECTSTART** *[width],[height]*

此命令和 START 命令类似，但允许棋盘为矩形棋盘，如果宽和高一样客户端应该使用 START 指令。
示例：

```
The manager sends:

RECTSTART 30,20

The brain answers:

OK - parameters are good

ERROR message - rectangular board is not supported or other error
```

- **RESTART**

该指令是在比赛结束或终止之后使用的，无参数。收到该指令后按照棋盘的宽高参数不变，重新初始化棋盘。
如果引擎回复 OK ,则之后的通信和 START 指令一样;
如果回复 UNKNOWN ，客户端则发送 END 指令然后重新加载引擎。
示例：

```
The manager sends:

RESTART

The brain answers:

OK
```

- **TAKEBACK** *[X]，[Y]*

此指令用于执行撤销上一步（undo）操作的，引擎撤去 [X，Y] 坐标上的棋子并回复 OK
示例：

```
The manager sends:

TAKEBACK 9,10

The brain answers:

OK
```

- **PLAY** *[X],[Y]*

该指令只用于客户端用于回复 SUGGEST 指令的，该指令让引擎执行在 [X]，[Y] 这一位置下子。
期望回复： 坐标位置，以逗号分割坐标。
如果引擎不想执行这一步可以发送其他坐标给客户端（不建议）
示例：

```
The brain has sent:

SUGGEST 10,10
```

```
The manager sends:

PLAY 12,10

The brain moves onto 12,10 and answers:

12,10
```

3、引擎发送的指令

客户端必须能够响应这些指令

- **UNKNOWN** *[error message]*

引擎如果接收到未知指令或者未实现指令则回复该指令。参数是用于客户端展示信息的。

- **ERROR** *[error message]*

当引擎接收到一些已知指令但无法处理时回复该指令。例如内存限制太小，棋盘规格太大。

- **MESSAGE** *[message]*

引擎向客户端发送的消息，可用在相关的日志 Log 中。

- **DEBUG** *[message]*

和 MESSAGE 指令类似，不过该指令只对引擎开发者有用，在 Gomocup 比赛中是不会显示出来的。
示例：

```
The manager sends:

TURN 10,15

The brain answers:

DEBUG The most promising move now is [10,14] alfa=10025 beta=8641

DEBUG The most promising move now is [11,14] alfa=10125 beta=8641

MESSAGE I will be the winner

10,16
```

- **SUGGEST** *[X]，[Y]*

引擎使用 SUGGEST[X]，[Y]代替[X]，[Y]回复 TURN、BEGIN 等相关指令，同时不改变棋盘内部状态。但客户段可能会忽略该指令并强制其他走子。
客户端可能回复的指令是 PLAY 或者 END。
在 Gomocup 比赛中客户端总是执行 SUGGEST 的走子。
大多数引擎不使用 SUGGEST 指令。

四、遵循通用协议的引擎正常工作流程

1. 载入引擎后，打开两个通信管道,然后等待客户端指令，
2. 引擎接到客户端发来 START 指令，初始化棋盘，如果成功则返回 OK，否则 ERROR
3. 当客户端接到 OK 的回复时，客户端会通过 INFO 指令发送一些限制信息到引擎，引擎处理但并不回复。
4. 如果引擎下的是整个棋局的第一步，客户端会发送 BEGIN 指令，否则发 TURN 指令，之后客户端等待引擎的回复。
5. 引擎通过思考回复下子的坐标给客户端。
6. 客户端会对回复的坐标进行检查，判断是否已决出胜负，是否是合法坐标，并执行相关操作。
7. 客户端切换引擎，执行第 4 步，并循环，直到有一方胜出。
8. 如果有一方胜出，记录并显示结果，客户端会判断是否继续新棋局，如果设置中设置了连续棋局，客户端会重开一个新棋局。

** *一下是一小段日志 Log：* **

```
______Process started______

START 20

OK

INFO max_memory 83886080

INFO timeout_match 180000

INFO timeout_turn 5000

INFO game_type 0

INFO rule 0

INFO time_left 179955

INFO folder D:\AI

TURN 12,6

11,5

INFO time_left 179930
```

```
TURN 7,3

7,2

INFO time_left 179890

TURN 7,6

7,7
```

五、协议实现分析

(一)客户端协议实现

由于本次实验已提供客户端可执行文件，这里忽略客服端协议实现介绍。

(二)引擎端协议实现

1. 引擎端协议控制逻辑在 pisqpipe.cpp 文件中，该文件包括指令的读取，指令响应逻辑控制，指令管道回复等功能，具体请看源码。
2. 在开发引擎时必须实现以下函数,以实现引擎功能：

   `brain_init, brain_restart, brain_turn, brain_my,`

   `brain_opponents, brain_block, brain_takeback, brain_end`

   其中最关键的思考功能在 brain_turn 函数中实现。
3. 本次实验在客户端设置需进行禁手检查。客户端通过 INFO 指令的 info_fb_check 字段通知引擎比赛需进行禁手检查。引擎端协议需对禁手功能的 INFO 指令进行响应，在引擎开发过程中先手方应该针对禁手功能进行处理。

- 增加 info_fb_check 字段记录是否进行禁手检查
- 在对 INFO 指令响应中增加对客户端发过来的 fb_check 关键字的响应

```
/** do command cmd */

static void do_command()

{

...

  if((param=get_cmd_param("info",cmd))!=0) {
```

```
    if((info=get_cmd_param("max_memory",param))!=0) info_max_memory=atoi(in
fo);

    if((info=get_cmd_param("timeout_match",param))!=0) info_timeout_match=a
toi(info);

    if((info=get_cmd_param("timeout_turn",param))!=0) info_timeout_turn=ato
i(info);

    if((info=get_cmd_param("time_left",param))!=0) info_time_left=atoi(inf
o);

    if((info=get_cmd_param("game_type",param))!=0) info_game_type=atoi(inf
o);

    if((info=get_cmd_param("rule",param))!=0){ e=atoi(info); info_exact5=e&1;
 info_continuous=(e>>1)&1; }

    if((info=get_cmd_param("folder",param))!=0) strncpy(dataFolder,info,siz
eof(dataFolder)-1);

    if((info=get_cmd_param("fb_check",param))!=0) info_fb_check = atoi(inf
o);

        ...
```