Кветка

Программа для анализа шахматных партий

ГЛАВНАЯ

О ПРОГРАММЕ

ИСТОРИЯ ИЗМЕНЕНИЙ

СКАЧАТЬ!

РУКОВОДСТВО

UCI-ДВИЖКИ

ДОКУМЕНТАЦИЯ

ГОСТЕВАЯ КНИГА

ПОДДЕРЖАТЬ

ОПИСАНИЕ ФОРМАТА UCI.

Формат UCI (Universal Chess Interface) используется для взаимодействия шахматных движков с их графическими оболочками. Шахматный движок представляет собой обычное приложение (чаще всего консольное). Оно получает различные команды по стандартному устройству ввода (stdin), как-то их обрабатывает и результат подаёт на стандартное устройство вывода (stdout). Таким образом, теоретически движок можно использовать и вручную: вводить команды на клавиатуре и смотреть, какие результаты будут выводиться. Однако это достаточно громоздко, да и требует хорошего знания UCI. Чтобы всё эту работу упростить, используются графические оболочки (GUI), которые общение с движком берут на себя и представляют все необходимые данные в удобном виде. Формат UCI описывает, какие команды движок может получить на вход, и как он должен на них реагировать.

ОБЩИЕ ПОЛОЖЕНИЯ.

1. После запуска движок должен войти в состояние ожидания команд. Он не должен начинать обдумывание без соответствующей команды.

2. Чтобы запуск происходил побыстрее, движок должен дождаться команд **"isready"** или **"setoption"** перед тем, как начать установку своих внутренних параметров. *Далеко не все движки придерживаются этого правила. Например Rybka 3 при запуске производит инициализацию каких-то своих параметров, в течение которой ввод команд оказывается невозможным.*

3. Движок должен быть готовым принимать команды на обработку в любое время, даже во время обдумывания.

4. Все передаваемые движку команды должны заканчиваться переводом строки ("\n"). Символом "\n" должна оканчиваться и каждая строка на выходе движка. В разных системах "\n" может кодироваться разными символами (0x0A, 0x0A0D и т.п.). *Было бы неплохо, если бы движки могли воспринимать разные варианты перевода строки, однако, к сожалению, мало кто это умеет.*

5. Между отдельными словами в командах можно вставлять любое число пробелов и символов табуляции.

6. Если на входе появляется команда, неизвестная движку, то он должен просто проигнорировать её и пытаться обработать строку так, будто команда эта отсутствует. То же самое касается и известных команд, появившихся в неположенном месте.

ПРИНЦИП РАБОТЫ ДВИЖКА.

Для того, чтобы движок начал обдумывание позиции, используется команда **"go"**. Однако сначала нужно подготовить позицию, которую движок будет анализировать. Это делается с помощью команды **"position"**. В принципе, эти две команды --- основные при работе с движком.

Во время обдумывания движок должен периодически выдавать результаты своей работы: рекомендуемые ходы, глубину обдумывания, затраченное на него время и т.д. (подробнее об этом см. далее). Чтобы остановить этот процесс, используется команда **"stop"**.

Чтобы убедиться, что запущенное приложение --- действительно UCI движок, рекомендуется первой командой ввести **"uci"**. В этом случае движок должен перейти в режим UCI, идентифицировать себя, передать информацию о поддерживаемых настройках и в конце выдать **"uciok"**. По последней строчке GUI и должна определять, что перед ним действительно UCI-движок.

ФОРМАТ ХОДОВ.

В UCI каждый ход описывается начальной и конечной координатой перемещаемой фигуры. Например: e2e4, b8c6, e1g1 (для рокировки). При превращении пешки в конце приписывается фигура, в которую она превратилась (n --- конь, b --- слон, r --- ладья, q --- королева). Например: e7e8q.

Пустой ход должен передаваться от движка к GUI как 0000. *Хотя большинство движков этого не делают. Рыбка, например, выдаёт a1a1, Fruit вообще ничего не выдаёт.*

ОПИСАНИЯ КОМАНД ДЛЯ ДВИЖКА.

1. **uci**
   Говорит движку использовать формат uci. Ожидается, что эта команда будет идти первой, сразу же после загрузки движка, чтобы перевести его в uci-режим.

   После получения команды "**uci**" движок должен идентифицировать себя с помощью команды **"id"** и затем послать GUI набор команд **"option"**, чтобы сообщить ему, какие настройки он поддерживает. В конце движок должен выдать команду **"uciok"**. Если в течение некоторого времени эта команда не появилась, то GUI следует закрыть приложение.

2. **debug** [on | off]
   Включает (on) или выключает (off) режим отладки движка.

   В режиме отладки движок должен посылать для GUI дополнительную отладочную информацию, например, команды **"info string"**.

   По умолчанию отладочный режим должен быть выключен. Эту команду можно ввести в любое время, даже во время обдумывания.

3. **isready**
   Посылается движку, чтобы выяснить, готов ли он к работе. Эту команду рекомендуется посылать, например, после других команд, требующих длительное время для своего выполнения. Она также может быть использована для проверки, не завис ли движок.

   После ввода команды движок обязан ответить **"readyok"**, как только он будет готов. В режиме обдумывания он также обязан немедленно ответить **"readyok"**, не прекращая при этом анализ.

4. **setoption name** #id [**value** #x]
   Посылается движку, чтобы изменить какой-то его внутренний параметр. Точнее, движок должен установить параметр с именем #id установить в значение #x.

   Некоторые стандартные имена #id и значения #x параметров будут даны позже при описании сообщения **"option"**. Кроме того, у движков могут быть свои собственные параметры. #id и #x могут состоять из нескольких слов, они не должны быть чувствительным к регистру и не должны содержать слов "name" и "value", чтобы не сбивать движок с толку. На каждый параметр должна выделяться одна строка. Эта команда может посылаться только когда движок находится в режиме ожидания.

5. **register**
   Используется для регистрации движка или для того, чтобы сообщить, что регистрация будет произведена позже. Эту команду обязательно необходимо послать, если при запуске движок выдал "**registration error**".

   Далее должно следовать одно из следующих слов:

   - **later**
     Пользователь пока не хочет регистрировать движок.

   - **name** #x
     Движок должен быть зарегистрирован на имя #x.

   - **code** #y
     Движок должен быть зарегистрирован кодом #y.
     Например: "register name DimaBest code 123456789".

6. **ucinewgame**
   Посылается движку, если анализ следующей позиции (идущий после команд "**position**" и **"go"**) будет производиться из другой партии.

   Движок не должен сильно полагаться на эту команду, поскольку GUI может её и не поддерживать. Если GUI не послал "**ucinewgame**" перед первой командой "**position**", то движку следует действовать так, как будто эта команда не поддерживается.

   Так как реакция на "

   ucinewgame" может занять некоторое время, рекомендуется после этой команды посылать **"isready"**, чтобы дождаться, пока движок не будет готов продолжать работу.

7. **position** [**fen** #fenstring | **startpos**] **moves** #move1 .... #movei
   Устанавливает новую позицию на внутренней доске. Именно эта позиция в дальнейшем будет анализироваться.

   Позиция устанавливается следующим образом: сначала на внутренней доске устанавливается начальная позиция, а затем проигрываются ходы #move1, #move2, ..., #movei. Начальная позиция либо стандартна (если записано "**startpos**"), либо устанавливается из #fenstring, описанной в формате fen (если записано **fen** #fenstring).

   Например: "position startpos moves e2e4 e7e5 b1c3"

   *Зачастую слово "**startpos**" можно опустить. Однако GUI лучше на это не полагаться.*

8. **go**
   Запускает анализ текущей позиции, установленной с помощью команды "**position**". За этой командой должно следовать несколько других команд. Все они пишутся в одной строке. Если какая-то из этих команд отсутствует, то движок должен проинтерпретировать её значение так, чтобы оно не влияло на анализ.

   Команды:

   - **searchmoves** #move1 .... #movei
     Ограничивает анализ только указанными ходами #move1, #move2, ..., #movei.
     Остальные возможные ходы исследоваться не должны.

   - **ponder**
     Начать исследование в режиме ожидания хода противника. В этом случае последний ход, записанный в команде "**position**" должен рассматриваться, как вероятный ход противника, над которым следует подумать.
     В принципе, движок не обязан анализировать только этот ход. Однако если он будет рассматривать и другие ходы, то ему не следует их подавать на вывод, чтобы не сбивать с толку GUI.
     Если в режиме ожидания хода противника последует команда **"ponderhit"**, это будет означать, что противник сделал тот ход, который и предполагалось.
     Движок не должен выходить из этого режима, даже если он видит мат.

   - **wtime** #x
     У белых на часах осталось #x миллисекунд.

   - **btime** #x
     У чёрных на часах осталось #x миллисекунд.

- **winc** #x
  За каждый ход белым прибавляется #x миллисекунд ко времени.

- **binc** #x
  За каждый ход чёрным прибавляется #x миллисекунд ко времени.

- **movestogo** #x
  До следующего контроля времени осталось #x ходов.
  Эту команду следует посылать только в случае x > 0. Если эта команда отсутствует, но присутствуют "**wtime**" и "**btime**", то время которое есть сейчас у игрока, дано ему до конца игры, и никаких добавлений (каждый ход или по контролю) не будет (sudden death).

- **depth** #x
  Исследовать позицию только до глубины в #x полуходов.

- **nodes** #x
  Исследовать только #x вероятных продолжений партии.

- **mate** #x
  Искать мат в #x ходов.

- **movetime** #x
  Обдумывать позицию в течение ровно #x миллисекунд. *Реально же, тут уж на сколько точности у системы хватит. Например, стандартные средства Windows 98 позволяют измерять время с точностью где-то 1/18 сек. В Windows XP --- 10 мсек.*

- **infinite**
  Исследовать позицию до тех пор, пока не встретится команда "**stop**".

9. **stop**
   После этой команды движок должен немедленно остановить анализ позиции.

   При завершении обдумывания должно быть выведено сообщение **"bestmove"** и, возможно, **"ponder"**.

10. **ponderhit**
    Означает, что противник походил именно так, как и предполагалось.

    Эта команда должна посылаться только в том случае, если движок находился в режиме ожидания хода противника (включается командой **"ponder"**). После этого движок должен продолжить анализ позиции, только уже в обычном режиме.

11. **quit**
    Немедленно выйти из программы.

ОПИСАНИЕ ВЫВОДА ДВИЖКА.

Вывод движка должен начинаться одним из следующих слов:

1. **id**
   Должно посылаться сразу после получения команды **"uci"**. Далее следуют:
   - **name** #x
     Выводит название движка.
     Например: "id name MyEngine 1.8"

   - **author** #x
     Выводит автора движка.
     Например: "id author Vasya Pupkin"

2. **uciok**
   Должно посылаться после строк с "**id**", и, возможно, с **"option"**. Гооврит GUI, что движок послал всю имеющуюся информацию и готов к работе в режиме uci.

3. **readyok**
   Должно посылаться в ответ на команду **"isready"** после того, как движок завершил выполнение всех задач, посланных ему ранее. Тем самым он сообщает, что готов принимать новые команды.

4. **bestmove** #move1 [**ponder** #move2]
   Означает, что движок закончил анализ позиции и рекомендует сделать в ней ход #move1.

   Движок также может вывести следующий ход, который он считает в партии наиболее вероятным. В этом случае выводится "**ponder** #move2". Движок не должен автоматически переходить в режим обдумывания этого хода.

   Эта строчка должна посылаться всегда, когда движок заканчивает анализ позиции. Даже в режиме ожидания хода противника, если была введена команда **"stop"**. Таким образом, на каждую команду **"go"** должна должна быть выведена "**bestmove**".

5. **copyprotection**
   Необходимо для движков, защищённых от копирования. После **"uciok"** может сообщить GUI, что сейчас он проверит защиту от копирования. Это делается с помощью строки "**copyprotection checking**".

   Если проверка проведена успешно, то необходимо вывести "**copyprotection ok**", в противном случае необходимо вывести "**copyprotection error**". В случае ошибки, движок не будет функционировать нормально, но он не должен при этом автоматически завершать работу.

   Если GUI встретит строку "**copyprotection error**", То ему не следует использовать этот движок, а вместо этого вывести сообщение об ошибке.

6. **registration**
   Необходимо для движков, требующих имя пользователя и регистрационный код для полноценной работы.

   По аналогии с "**copyprotection**" после строки **"uciok"** движок должен послать "**registration checking**", и далее "**registration ok**" или "**registration error**", в зависимости от успешности проверки.

   Также после каждой попытки зарегистрироваться (командой **"register"**), движок должен отвечать строками "**registration checking**" и далее "**registration ok**" или "**registration error**".

   В отличие от "**copyprotection**" после сообщения об ошибке регистрации движком можно продолжать пользоваться. Но GUI следует предупредить пользователя, что движок не зарегистрирован надлежащим образом и, возможно, не использует все имеющиеся возможности. Также GUI следует предоставить пользователю диалог для регистрации движка. GUI обязан ответить движку командой **"register"** после каждого сообщения "**registration error**".

7. **info**
   Означает, что движок передаёт информацию для GUI. Оно должно выводиться каждый раз, когда некоторая её часть изменилась.

   За одним словом "**info**" может следовать несколько сообщений для GUI. Все сообщения, привязанные к **"pv"**, должны следовать в одной строке. За "**info**" могут следовать следующие слова:

   - **depth** #x
     Означает, что текущая глубина анализа --- #x полуходов.

   - **seldepth** #x
     Означает, что текущая глубина анализа избранных ходов --- #x полуходов.
     Если движок посылает "**seldepth**", то в этой же строчке он обязан послать и "**depth**".

   - **time** #x
     Время исследования --- #x мсек. Должно выводиться вместе с "**pv**".

   - **nodes** #x
     обработано #x позиций. Движок должен регулярно выдавать это сообщение.

   - **pv** #move1 ... #movei
     Наилучшее продолжение партии.

   - **multipv** #num
     Это сообщение выдаётся, если выбран режим **multipv**. В этом случае оно должно выводиться вместе с "**pv**". Для лучшего хода выводится "**multipv** 1", для следующего --- "**multipv** #2" и т.д.
     Если выбран режим выводить k рекомендованных ходов, то все k вариантов должны выводиться подряд в k строках.

   - **score**
     Оценка позиции после рекомендованного хода. Должно выводиться вместе с **"pv"**.
     Возможные продолжения:
     - **cp** #x
       Оценка в десятых долях пешки. Положительное значение #x означает, что позиция лучше у того, чей сейчас ход. Отрицательное --- у его соперника.

     - **mate** #y
       Мат в #y ходов. Положительное значение #y означает, что мат ставит тот, чей сейчас ход. Отрицательное --- его соперник.

     - **lowerbound**
       Приводится нижняя оценка позиции.

     - **upperbound**
       приводится верхняя оценка позиции.

   - **currmove** #move
     В данный момент движок исследует ход #move.

   - **currmovenumber** #x
     Сейчас анализируется ход №#x. Для первого хода #x должно быть равно 1, а не 0.

   - **hashfull** #x
     Хеш заполнен на #x промилле (0.1%). Движок должен регулярно выдавать это сообщение.

   - **nps** #x
     За секунду было обработано #x возможных продолжений. Движок должен регулярно выдавать это сообщение.

   - **tbhits** #x
     В эндшпильных базах было найдено #x позиций.

   - **sbhits** #x
     В эндшпильных базах Шредера (shredder endgame databases) было найдено #x позиций.

   - **cpuload** #x

Процессор загружен на #x промилле (0.1%).

- **string** #str
  Посылается произвольная строка. В качестве #str должен быть проинтерпретирован весь текст до конца строки.

- **refutation** #move1 #move2 ... #movei
  ход #move1 опровергается последовательностью ходов #move2, #move3, ..., #movei, где i --- любое число >= 1.
  Пример: после анализа хода d1h5 движок может вывести "info refutation d1h5 g6h5" в случае если g6h5 --- наилучший ответ на d1h5 либо g6h5 опровергает ход d1h5. Если для хода d1h5 опровержений не найдено, то движок может вывести "info refutation d1h5".
  Движок должен выводить эту информацию только в том случае, если опция **"UCI_ShowRefutations"** установлена в значение true.

- **currline** #cpunr #move1 ... #movei
  В данный момент анализируется последовательность ходов #move1, #move2, ..., #movei. #cpunr указывает на номер процессора, анализирующего данную последовательность (#cpunr = 1, 2, 3, ...). Если работает только один процессор, то #cpunr может в сообщении отсутствовать.
  Если используется k процессоров, то все k строк с сообщениями "**currline**" должны располагаться подряд.
  Движок должен выводить эту информацию только в случае, когда опция **"UCI_ShowCurrLine"** установлена в значение true.

8. **option**
   Говорит GUI о том, какой параметр может быть в движке изменён.

   Строки с сообщениями "**option**" должны посылаться в ответ на команду **"uci"** после сообщений **"id"** в случае, если какой-то параметр может быть изменён. Чтобы изменить его, GUI должен послать команду **"setoption"**. В одной строке может содержаться информация только об одном параметре.

   Далее в строке должны следовать комбинации из следующих слов (все допустимые комбинации будут приведены в примерах):

   - **name** #id
     Параметр имеет имя #id.
     Некоторые из параметров имеют стандартизированные имена. Большинство из них, кроме первых 6-ти, начинаются с приставки "UCI_". Эта приставка сама по себе означает, что имя зарезервировано. Если GUI встретит неизвестное ему имя с этой приставкой, то оно должно его проигнорировать.
     Список стандартных параметров:
     - #id = **Hash**, тип **spin**
       объём памяти в МБ для хеш-таблиц.
       Этот параметр следует установить первым с помощью **"setoption"** после запуска движка. Он должен поддерживаться всеми движками.
       По умолчанию движок должен выделять очень мало памяти для своих хеш-таблиц.

     - #id = **NalimovPath**, тип **string**
       Путь к эндшпильным базам в сжатом формате Налимова (Nalimov compressed format).
       Можно указывать несколько путей, разделённых символом ";".

     - #id = **NalimovCache**, тип **spin**
       Объём памяти в МБ под кэш для таблиц Налимова (nalimov table bases).

     - #id = **Ponder**, тип **check**
       Означает, что движок поддерживает режим обдумывания партии в ожидании хода противника.
       В принципе, GUI может послать команду ponder в независимости, поддерживается этот режим или нет, так что к этому надо быть готовым. Движок не должен начинать анализ в этом режиме сам по себе. Этот параметр нужен только потому, что в режиме анализа партии в ожидании хода противника движок может скорректировать своё управление временем.

     - #id = **OwnBook**, тип **check**
       Означает, что движок имеет свою собственную дебютную книгу, к которой он имеет доступ. Если установлен этот параметр, то движок черпает информацию из своей дебютной книги и будет обрабатывать только ходы, выходящие за её пределы. Если параметр установить в false, то движок не должен обращаться к своей дебютной книге.

     - #id = **MultiPV**, тип **spin**
       Движок поддерживает возможность вывода нескольких рекомендуемых ходов. Значение по умолчанию --- 1.

     - #id = **UCI_ShowCurrLine**, тип **check**
       Движок может возвращать информацию о текущей последовательности ходов, над которой он думает. (см. **"info currline"**). Значение по умолчанию --- false.

     - #id = **UCI_ShowRefutations**, тип **check**
       Движок может показывать ход и его опровержения (см. **"info refutation"**). Значение по умолчанию --- false.

     - #id = **UCI_LimitStrength**, тип **check**
       Движок может ограничивать свою силу до конкретного значения Elo. Значение

по умолчанию --- false.
Этот параметр должен быть реализован вместе с "**UCI_Elo**".

- #id = **UCI_Elo**, тип **spin**
  Движок может ограничивать свою силу в Elo в пределах указанного интервала. Если "**UCI_LimitStrength**" установлен в false, то это значение должно игнорироваться. Если же он установлен в true, то движок должен анализировать ходы в соответствие с указанной силой.
  Этот параметр должен быть реализован вместе с "**UCI_LimitStrength**".

- #id = **UCI_AnalyseMode**, тип **check**
  Означает, что движок ведёт себя по-разному при анализе партии и при игре в шахматы. Например, при игре он может самообучаться.
  Устанавливается в false, если движок играет в шахматы. В противном случае устанавливается в true.

- #id = **UCI_Opponent**, тип **string**
  С помощью этого параметра GUI может послать движку имя, титул, Elo игрока, а также играет ли движок с человеком или с компьютером.
  Формат строки должен быть следующим:
  [GM|IM|FM|WGM|WIM|none] [#elo|none] [computer|human] #name
  Примеры:
  "setoption name UCI_Opponent value GM 2800 human Gary Kasparov"
  "setoption name UCI_Opponent value none none computer Shredder"

- #id = **UCI_EngineAbout**, тип **string**
  С помощью этого параметра движок сообщает GUI некоторую информацию о себе, например, лицензионное соглашение. Обычно изменять его с помощью **"setoption"** не имеет смысла.
  Пример:
  "option name UCI_EngineAbout type string default MyEngine By Vasya Pupkin"

- #id = **UCI_ShredderbasesPath**, тип **string**
  Либо указывает путь на каталог, содержащий эндшпильные базы Шреддера (Shredder endgame databases), либо на конкретный файл с эндшпильными базами Шреддера.

- #id = **UCI_SetPositionValue**, тип **string**
  GUI может изменить этот параметр, если хочет установить конкретный рейтинг для определённой позиции в десятых долях пешки с точки зрения белых.
  Строка должна быть в одном из следующих форматов:
  #value + #fen | **clear** + #fen | **clearall**
  В первом случае рейтинг позиции #fen, записанной в формате FEN, должен быть установлен в #value. Во втором случае рейтинг позиции #fen должен устанавливаться движком. В третьем случае, рейтинг любой позиции должен устанавливаться движком.

- **type** #t
  Параметр имеет тип #t. Всего имеется 5 различных типов для параметров:
  - **check**
    Логический параметр. Может принимать значения true и false. (в диалоге должен оформляться в виде checkbox).
  - **spin**
    целочисленный параметр, изменяющийся в заданных пределах. (В диалоге должен оформляться в виде ползунка).
  - **combo**
    Параметр, принимающий значения из множества предустановленных строковых значений. (В диалоге должен оформляться в виде группы из combo box.)
  - **button**
    Параметр, не принимающий значений. С его помощью GUI может отправлять соответствующую команду движку. (В диалоге должен оформляться в виде кнопки).
  - **string**
    Строковый параметр. (В диалоге должен оформляться в виде поля редактирования).
    Пустая строка оформляется значением "< empty >".

- **default** #x
  Значение данного параметра по умолчанию --- #x.

- **min** #x
  Минимальное значение данного параметра --- #x.

- **max** #x
  Максимальное значение данного параметра --- #x.

- **var** #x
  Возможное строковое значение этого параметра --- #x.

Примеры: приведём 5 примеров, соответствующих 5-ти возможным типам параметра.
"option name Nullmove type check default true"

"option name Selectivity type spin default 2 min 0 max 4"
"option name Style type combo default Normal var Solid var Normal var Risky"
"option name NalimovPath type string default c:\"
"option name Clear Hash type button"

ПРИМЕР РАБОТЫ С UCI.

Рассмотрим, как может выглядеть общение между движком и GUI.

=>: команда --- GUI
сообщение --- движок.

//Сначала говорим движку переключиться в режим UCI
=>: uci

//Движок идентифицирует себя
id name Shredder
id author Stefan MK

//Движок посылает параметры, которые можно изменить
//Движок может установить размер хеша от 1 до 128 МБ
option name Hash type spin default 1 min 1 max 128

//Движок поддерживает эндшпильные таблицы Налимова
option name NalimovPath type string default < empty>
option name NalimovCache type spin default 1 min 1 max 32

//Движок может отключить пустой ход и установить стиль игры
option name Nullmove type check default true
option name Style type combo default Normal var Solid var Normal var Risky

//Движок выслал все параметры и готов к работе
uciok

//Отметим, что GUI может послать команду **"quit"** уже сейчас,
//если, например, он всего лишь хотел узнать, что умеет этот движок.
//Таким образом, движку не следует инициализировать
//свои внутренние параметры до этого момента. *Хотя зачастую они это делают.*
//Теперь GUI устанавливает в движке некоторые из параметров
//Установить объём хеша в 32 МБ
=>: setoption name Hash value 32

//Инициализировать эндшпильные таблицы
=>: setoption name NalimovCache value 1
=>: setoption name NalimovPath value d:\tb;c\tb

//Ждём, пока движок не завершит инициализацию
=>: isready

//Движок завершил инициализацию и готов продолжать
readyok

// Теперь мы готовы к работе
// Если GUI это поддерживает, сообщим движку,
//что он будет анализировать игру, которую ещё не анализировал ранее.
=>: ucinewgame

// Если движок поддерживает параметр **"UCI_AnalyseMode"**
//и мы собираемся именно анализировать партию,
//то GUI должен установить **"UCI_AnalyseMode"** в true.
=>: setoption name UCI_AnalyseMode value true

//Говорим движку начать бесконечный анализ позиции после ходов 1.e4 e5
=>: position startpos moves e2e4 e7e5
=>: go infinite

//Движок начинает посылать информацию о ходе анализа
//(приведём только некоторые из возможных вариантов вывода)

info depth 1 seldepth 0
info score cp 13 depth 1 nodes 13 time 15 pv f1b5
info depth 2 seldepth 2
info nps 15937
info score cp 14 depth 2 nodes 255 time 15 pv f1c4 f8c5
info depth 2 seldepth 7 nodes 255
info depth 3 seldepth 7
info nps 26437
info score cp 20 depth 3 nodes 423 time 15 pv f1c4 g8f6 b1c3
info nps 41562
....

//Мы получили достаточно информации и просим движок остановиться
=>: stop

//Движок прекращает анализ и посылает сообщение **"bestmove"**, которое должно появиться после каждой команды **"go"**, чтобы передать GUI, что движок готов продолжать.
bestmove g1f3 ponder d8f6

Вольный перевод с английского спецификации UCI
Автор перевода --- Бодягин Дмитрий