Universal Chess Interface (UCI)

This is the description of a new interface between a chess engine and a graphical user interface called UCI. It was designed by Rudolf Huber and Stefan Meyer-Kahlen and is used in the chess engines SOS and Shredder as well as in the Shredder graphical user interface. The UCI interface is free to use for everyone, so you can use it in your own program *without any licence fees or restrictions*.

The interface is similar to Winboard, keeping its easiness but eliminating the disadvantages of it. It should be not much work to adept an chess engine to UCI especially if it's already supporting winboard. It is also possible to support both UCI and winboard in the same exe file.

What are the advantages of UCI compared to Winboard?

- All engine options can be modified within the graphical user interface so there is no need to deal with ini files.
- Much better capabilities to display search information of the engine,
- Definition of a principal variation is included,
- It's more robust, the GUI always knows exactly what the engine is doing.
- It's supporting multi variation mode,
- Support for endgame tablebases
- Flexible time controls,
- The engine can identify itself
- UCI is supporting a copy protection mechanism (for the professionals)

The advantages to other engine interfaces are:

- It's free
- It's easy to use
- It's easy to extent
- It's independent of the operating system
- It's capable of network play

Below is the "formal" definition of the engine interface, if you have any more question concerning UCI I will try to answer them.

Stefan Meyer-Kahlen

Description of the Universal Chess Interface (UCI)

The specification is independend of the operationg system. For Windows, the engine is a normal exe file, either a console or "real" windows application.

- all communication is done via standard input and output with plain text commands,

- The engine should boot and wait for input from the GUI, the engine should wait for the "isready" or "setoption" command to set up its internal parameters as the boot process should be as quick as possible.

- the engine must always be able to process input from stdin, even while thinking.

- all command strings the engine receives will end with '\n', also all commands the GUI receives should end with '\n',

- The engine will always be in forced mode which means it should never start calculating or pondering without receiving a "go" command first.

- The engine should never execute a move on its internal chess board without being asked to do so by the GUI, e.g. the engine should not execute the best move after a search.

- Before the engine is asked to search on a position, there will always be a position command to tell the engine about the current position.

- all the opening book handling is done by the GUI, but there is an option for the engine to ues its own book ("OwnBook" option, see below)

Move format:

The move format is in long algebraic notation. Examples: e2e4, e7e5, e1g1 (white short castling), e7e8q (for promotion)

GUI to engine:

These are all the command the engine gets from the interface.

uci

tell engine to use the uci (universal chess interface),   this will be send once as a first command after program bootto tell the engine to switch to uci mode.After receiving the uci command the engine must identify itself with the "id" commmandand sent the "option" commands to tell the GUI which engine settings the engine supports if any. After that the engine should sent "uciok" to ackknowledge the uci mode. If no uciok is sent within a certain time period, the engine task will be killed by the GUI.

debug [on | off]

switch the debug mode of the engine on and off. In debug mode the engine should sent additonal infos to the GUI, e.g. with the "info string" command, to help debugging, e.g. the commands it the engine has reveived etc. This mode should be switched off per default and this command can be sent any time, also when the engine is thinking.

isready

this is used to synchronize the engine with the GUI. When the GUI has sent a command ormultiple commands that can take some time to complete, this command can be used to wait for the engine to be ready again or to ping the engine to find out if it is still alive. E.g. this should be sent after setting the path to the tablebases as this can take some time. This command is also required once before the engine is asked to do any search to wait for the engine to finish initializing. This command must always be answered with "readyok"

setoption name <id> [value <x>]

this is sent to the engine when the user wants to change the internal paramters of the engine. For the "button" type no value is needed. One string will be sent for each paramter and this will only be sent when the engine is waiting. The name of the option in <id> should not be case sensitive!

Here are some strings for the example below:

```
"setoption name Nullmove value true\n"
"setoption name Selectivity value 3\n"
"setoption name Style value Risky\n"
"setoption name Clear Hash\n"
"setoption name NalimovPath value c:\chess\tb\4;c:\chess\tb\5\n"
```

position [fen <fenstring> | startpos] moves <move1> .... <movei>

set up the position described in fenstring on the internal board and play the moves on the internal chess board. if the game was played from the start postion the string "startpos" will be sent
Note: no "new" command is needed.

Go

start calculating on the current position There are a number of commands that can follow this command, all will be sent in the same string. If one command is not send its value should be interpreted as it would not influence the search.

searchmoves <move1> .... <movei>

restrict search to this moves only

ponder
start searching in pondering move.
Do not exit the search in ponder mode, even if it's mate!
This means that the last move sent in in the position string is the ponder move.
The engine can do what it wants to do, but after a "ponderhit" command
it should execute the suggested move to ponder on.

*** wtime <x>**
white has x msec left on the clock
*** btime <x>**
black has x msec left on the clock
*** winc <x>**
white increment per move in mseconds if x > 0
*** binc <x>**
black increment per move in mseconds if x > 0
*** movestogo <x>**
there are x moves to the next time control,
this will only be sent if x > 0,
if you don't get this and get the wtime and btime it's sudden death
*** depth <x>**
search x plies only.
*** nodes <x>**
search x nodes only,
*** mate <x>**
search for a mate in x moves
*** movetime <x>**
search exactly x mseconds
*** infinite**
search until the "stop" command.

Do not exit the search without being told so in this mode!

*** stop**

stop calculating as soon as possible,
don't forget the "bestmove" command when finishing the search!

*** ponderhit**

user has played the expected move. This will be sent if the engine is pondering on the same move the user has played. The engine should continue searching but switch from pondering to normal search.

*** quit**

quit the program as soon as possible

Engine to GUI:

*** id**

* name <x>
this must be sent after receiving the "uci" command to identify the engine,
e.g. "id name Shredder 5.0\n"
* author <x>
this must be sent after receiving the "uci" command to identify the engine,
e.g. "id author Stefan MK\n"

*** uciok**

Must be sent after the id and optional options to tell the GUI that the engine
has sent all infos and is ready in uci mode.

*** readyok**

This must be sent when the engine has received an "isready" command and has processed all input and is ready to accept new commands now.
It is usually sent after a command that can take some time to be able to wait for the engine,
but it can be used anytime, even when the engine is searching,
and must always be answered with "isready".

*** bestmove <move1> [ponder <move2>]**

the engine has stopped searching and found the move <move> best in this position. the engine can send the move it likes to ponder on. The engine must not start pondering automatically. this command must always be sent if the engine stops searching, also in pondering mode if there is a "stop" command, so for every "go" command a "bestmove" command is needed!

Directly before that the engine should send a final "info" command with the final search information, the the GUI has the complete statistics about the last search.

*** copyprotection**

this is needed for copyprotected engines. After the uciok command the engine can tell the GUI, that it will check the copy protection now. This is done by "copyprotection checking".
If the check is ok the engine should sent "copyprotection ok", otherwise "copyprotection error".
If there is an error the engine should not function properly but should not quit alone.

The code in the engine can look like this

```
      TellGUI("copyprotection checking\n");
         // ... check the copy protection here ...
         if(ok)
            TellGUI("copyprotection ok\n");
      else
         TellGUI("copyprotection error\n");
```

*** info**

the engine wants to send infos to the GUI. This should be done whenever one of the info has changed.
The engine can send only selected infos and multiple infos can be send with one info command,
e.g. "info currmove e2e4 currmovenumber 1".
Also all infos belonging to the pv should be sent together
e.g. "info depth 2 score cp 214 time 1242 nodes 2124 nps 34928 pv e2e4 e7e5 g1f3
I suggest to start sending "currmove" and "currmovenumber" only after one second to avoid traffic.

Additional info:

*** depth <x>**
search depth in plies

*** seldepth <x>**
selective search depth in plies,
if the engine sends seldepth there must also a "depth" be present in the same string.

*** time <x>**
the time searched in ms, this should be sent together with the pv.

*** nodes <x>**
x nodes searched, the engine should send this info regularly

*** pv <move1> ... <movei>**
the best line found

*** multipv <num>**
this for the multi pv mode.
for the best move/pv add "multipv 1" in the string when you send the pv.
in k-best mode always send all k variants in k strings together.

*** score**

- *** cp <x>**
  the score from the engine's point of view in centipawns
- *** mate <y>**
  mate in y moves
- *** lowerbound**
  the score is just a lower bound
- *** upperbound**
  the score is just an upper bound

*** currmove <move>**
currently searching this move

*** currmovenumber <x>**
currently searching move number x, for the first move x should be 1 not 0.

*** hashfull <x>**
the hash is x permill full, the engine should send this info regularly

*** nps <x>**
x nodes per second searched, the engine should send this info regularly

*** tbhits <x>**
x positons where found in the endgame table bases

*** cpuload <x>**
the cpu usage of the engine is x permill.

*** string <str>**

any string str which will be displayed be the engine,
if there is a string command the rest of the line will be interpreted as <str>.

*** option**

This command tells the GUI which parameters can be changed in the engine.
This should be sent once at engine startup after the "uci" and the "id" commands
if any parameter can be changed in the engine.
The GUI should parse this and build a dialog for the user to change the settings.
If the user wants to change some settings, the GUI will send a "setoption" command
to the engine.

For all allowed combinations see the example below, as some combinations of this
tokens don't make sense.

One string will be sent for each paramter.

*** name <id>**

The option has the name id.
This can be sent in any language, the GUI will take care of the translation.
Certain options have a fixed value for <id>, which means that the semantics
of this option is fixed

* <id> = Hash, type is spin
the value in MB for memory for hash tables can be changed,
this should be answered with the first "setoptions" command at
program boot
if the engine has sent the appropiate "option name Hash" command,
which should be supported by all engines!

So the engine should use a very small hash first as default.

* <id> = NalimovPath, type string
this is the path on the harddisk to the Nalimov compressed format.
Multiple directories can be concatenated with ";"

* <id> = NalimovCache, type spin
this is the size in MB for the cache for the nalimov table bases
These last two options should also be present in the initial options
exchange **dialog**
when the engine is booted if the engine supports it

* <id> = Ponder, type check
this means that the engine is able to ponder.
The GUI will send this whenever pondering is possible or not.
Note: The engine should not start pondering on its own if this is
enables, this option is only needed because the engine might change
its time management algorithm when pondering is allowed.

* <id> = OwnBook, type check
if this is set, the engine takes care of the opening book and the GUI
will never
execute a move out of its book for the engine.

* <id> = MultiPV, type spin
the engine supports multi best line or k-best mode. the default value
is 1

*** type <t>**

The option has type t.
There are 5 different types of options the engine can send
* check

```
				a checkbox that can either be true or false
			* spin
				a spin wheel that can be an integer in a certain range
			* combo
				a combo box that can have different predefined strings as a value
			* button
				a button that can be pressed to send a command to the engine
			* string
				a text field that has a string as a value,
				an empty string has the value "<empty>"
		* default <x>
			the default value of this parameter is x
		* min <x>
			the minimum value of this parameter is x
		* max <x>
			the maximum value of this parameter is x
		* var <x>
			a predefined value of this paramter is x

Example:   Here are 5 strings for each of the 5 possible types of options

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

Example:

This is how the communication when the engine boots can look like:

```
GUI     engine

// tell the engine to switch to UCI mode
uci

// engine identify
        id name Shredder 5
			id author Stefan MK

// engine sends the options it can change
// the engine can change the hash size from 1 to 128 MB
			option name Hash type spin default 1 min 1 max 128

// the engine supports Nalimov endgame tablebases
			option name NalimovPath type string name c:\
			option name NalimovCache type spin default 1 min 1 max 32

// the engine can switch off Nullmoves and set the playing style
		option name Nullmove type check default true
			option name Style type combo default Normal var Solid var Normal var Risky

// engine has sent all parameters and is ready
			uciok
```

```
// Note: here the GUI can already send a "quit" command if it just wants to find out
//       details about the engine, so the engine should not initialize its internal
//       parameters before here.
// now the GUI sets some values in the engine
// set hash to 32 MB
setoption name Hash value 32

// init tbs
setoption name NalimovCache value 1
setoption name NalimovPath value d:\tb;c\tb

// waiting for the engine to finish initializing
// this command and the answer is required here!
isready

// engine has finished setting up the internal values
                    readyok

// now we are ready to go
// tell the engine to search infinite from the start position after 1.e4 e5
position startpos moves e2e4 e7e5
go infinite

// the engine starts sending infos about the search to the GUI
// (only some examples are given)


                    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
                    ....

// here the user has seen enough and asks to stop the searching
stop

// the engine has finished searching and is sending the bestmove command
// which is needed for every "go" command sent to tell the GUI
// that the engine is ready again
                    bestmove g1f3 ponder d8f6
```