Subject: What about LUA plugin?
Posted by mirek on Tue, 22 Aug 2006 16:50:06 GMT
View Forum Message <> Reply to Message

Anobody to contribute?

Mirek

Subject: Re: What about LUA plugin?
Posted by fudadmin on Tue, 22 Aug 2006 19:28:45 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 22 August 2006 17:50Anobody to contribute?

Mirek

I've tried Lua. Lua is slow. I think slower than Ruby or similar. No good! Or at least, they are too much below my speed patience level
Python is faster but I don't like its syntax and its C base.
...The main reason of arilect.com and then why I joined U++ was ... interpreters things and my favourite interpreter Dialect. Also, not very known like U++. Even less. But very fast! And very nice syntax.

I made a very limited working version (UDialect?) with some of U++ Ctrls in February this year but then I had learn more U++ things and things like PHP, forums, agg, some more databases etc...

And the reason I was asking some questions about Esc speeds was that I want to make the opposite thing as well - a good interpreter embedded into U++...

So, anyway, I expect to publish it in the nearest future (2-3 months?...)

But at the moment I want to polish agg-svg-upp things (1 month).

Btw, why Lua?

If you are interested about Dialect:
(the first link doesn't work with opera)
http://secure.mtd-inc.com/dialect/
(and the project itself which is very very quiet. Unlike U++)
http://dialect.sourceforge.net/

Subject: Re: What about LUA plugin?
Posted by masu on Tue, 22 Aug 2006 21:01:04 GMT
View Forum Message <> Reply to Message

I like lua for its simplicity (clean and easy syntax, small code base, easy to extend and easy to use as an extension language).
Actually, I wondered why it never came into your mind using it as extension language instead of writing your own, Esc.
I started playing around with it a bit for using in upp, but up to now I have not got the time to go for it seriously. I hope I can do more on it soon.

Matthias

Subject: Re: What about LUA plugin?
Posted by unodgs on Tue, 22 Aug 2006 21:07:24 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Tue, 22 August 2006 15:28
Btw, why Lua?

Because it's very modern, flexible language and it has really small footprint. The library size is about 200-300kb. In my company we added sql interface to it and use it to write quick-fix apps.

Subject: Re: What about LUA plugin?
Posted by fudadmin on Tue, 22 Aug 2006 21:40:12 GMT
View Forum Message <> Reply to Message

unodgs wrote on Tue, 22 August 2006 22:07fudadmin wrote on Tue, 22 August 2006 15:28
Btw, why Lua?

Because it's very modern, flexible language and it has really small footprint. The library size is about 200-300kb. In my company we added sql interface to it and use it to write quick-fix apps.

Ok. Let's compete, then...

Subject: Re: What about LUA plugin?
Posted by mirek on Tue, 22 Aug 2006 22:13:52 GMT
View Forum Message <> Reply to Message

masu wrote on Tue, 22 August 2006 17:01Actually, I wondered why it never came into your mind using it as extension language instead of writing your own, Esc.

One of reasons is that we needed something more similar to C++ - to copy Paint methods into

.usc.

Another is that Esc is easily extensible (e.g. .usc), and C++ binding is quite easy.

Last but not least, inventing language is fun

Anyway, originally I have not planed to use it outside TheIDE.

Mirek

Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 14:16:16 GMT
View Forum Message <> Reply to Message

Hello,

interesting to hear that Lua was to slow.
From benches (http://shootout.alioth.debian.org/) it looks faster than other script languages, so I'd be happy to share real life experience.

What were you trying to do with it and which version of Lua have you been using and on which platform ? This could help understanding the limits of using it.

Well this is no news that interpreted languages are slower than compiled ones, however, you can profile and balance your application with good partitionning between C parts, and script parts (and if you are lucky, with hardware parts...). So that you can accelerate scripts algo with some C code, when needed...

Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 14:44:14 GMT
View Forum Message <> Reply to Message

A first reason I got interested in Lua, was that it was first intended to be a configuration language. So it is quite efficient for that. And far more better than using classical map files like that and still widely in use:
m.p.i = 0
m.p.s = "toto"
...

I now see that people are using more and more XML for that. And I'm quite puzzled by this:
- this is still a data only language, thus impossible to embed calculated settings, nor consistency

checks
- references needs to be managed by hand
- syntax is horribly verbose and thus obfuscating, what is the gain of ascii then?

By the way, does U++ provides a configuration API, or just rely upon XML and its serialization features ?

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 16:56:47 GMT
View Forum Message <> Reply to Message

The basic configuration system of U++ is extremly effective, fast and simple binary form, using "Serialize" method and streams.

Other options include using XML, possibly in Xmlize form, or CParser (if you need your config to look c-like).

Mirek

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 17:11:27 GMT
View Forum Message <> Reply to Message

P.S.:

reference/Serialize
reference/Xmlize
reference/CParser

Last one does not exactly deals with configuration, but shows how to use CParser. Similiary, you could create interpreter of your config file.

Mirek

Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 17:45:35 GMT
View Forum Message <> Reply to Message

Ok. And you shall add Esc for custom controls in layout editor.

Does serialization cope with endianness, word sizes? That is can you exchange configuration between a (not x86) linux and a x86 linux or win32?

Well drawback with binary configuration file is that they can be edited outside of tool itself... which can be boring if for some reason they are broken (like switch off of computer during saving, but i hope you at least do a two phases serialization to back up a little). Another drawback is that it is hard making a modular serialization: forcing to stream all at once.

Talking about "fast". I don't see it much as a requirement for a configuration with is quite a limited data space...

Well maybe a contribution of mine could be to share a lualize if I develop one.
Then I just wonder if there would be kind a uppforge.net for custom packages !

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 18:50:19 GMT
View Forum Message <> Reply to Message

thierry wrote on Sun, 10 September 2006 13:45Ok. And you shall add Esc for custom controls in layout editor.

No, I will not add Esc for custom controls. I have done so 2 years ago

Quote:
Does serialization cope with endianness, word sizes? That is can you exchange configuration between a (not x86) linux and a x86 linux or win32?

Yes. However, it is true that keeping binary config files compatible between versions can be a bit tricky, that is why I would recommend using something else for "protocolar" files (that is XMLize or custom text format and CParser).

Quote:
Then I just wonder if there would be kind a uppforge.net for custom packages !

I hope that in future, yes. At the moment, we are not as popular yet

Mirek

Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 21:47:38 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 10 September 2006 20:50thierry wrote on Sun, 10 September 2006 13:45Ok. And you shall add Esc for custom controls in layout editor.

No, I will not add Esc for custom controls. I have done so 2 years ago

Quote:
Then I just wonder if there would be kind a uppforge.net for custom packages !

I hope that in future, yes. At the moment, we are not as popular yet

Mirek

About Esc, I wasn't saying the intregation of Esc was a todo, but just mentionning it as yet another configuration mean of theIDE to add to the list xml, serialize.

I was more trying to put the emphasis on the fact that configuration means were multiplying. All with their pros and cons, then without deep design consideration about why using them other than marketing done around them.

But in the end they all tend to be used, thus asking more code to be written (even if small, this is still a bloat) and cores to be linked.

Given machine strength, most of the needs could be satisfied with one small core script language, like lua proves, but why not another as long as it can have a critical mass, or a database when data are larger.

Then why paying the effort of maintaining a script language where you don't want to maintain a database engine? And to which extent, this can predate the development of other nice features.

Don't take it for U++, this is more general consideration about users architecture decision, and a tool like U++ aims to address as many kind of users needs as possible. Including yours at first with your existent code base.

My only worry would be that serialization is more supported by external streaming operators in a more modular and extensible aspect oriented design.
something like:

```
#include <iostream>

struct XmlIO {
  template<class T>  XmlIO& operator()(const char* tag, T& v)
    { std::cout << "<" << tag << ">" << v << "<\\" << tag << ">"; return *this; }
};

#define SERIALIZABLE(Class)  template<class stream> friend stream& operator<<(stream&, Class&);
```

```
// End user code
class C {
  int v1, v2;
  SERIALIZABLE(Class);
public:
  C() : v1(12), v2(24) {}
};

template<> std::ostream& operator<<(std::ostream& s, C& c) { return s << c.v1 << std::endl <<
c.v2; }
template<> XmlIO& operator<<(XmlIO& xml, C& c)   { return xml("v1", c.v1)("v2", c.v2); }

int main() {
  C c;
  std::cout << c << std::endl;
  XmlIO s;
  s << c;
  return 0;
}
```

About a uppforge, why waiting high demand? Won't the offer create demand? At least, this shall only encourage sharing U++ code, thus giving more examples of U++ code. Maybe just letting a space where to put code snipset would be nice.

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 22:24:34 GMT
View Forum Message <> Reply to Message

Quote:
```
struct XmlIO {
  template<class T>  XmlIO& operator()(const char* tag, T& v)
    { std::cout << "<" << tag << ">" << v << "<\\" << tag << ">"; return *this; }
};
```

It is bidirectional, so cout does not make it, however I see the point...

Quote:
```
#define SERIALIZABLE(Class)  template<class stream> friend stream& operator<<(stream&,
Class&);

// End user code
class C {
  int v1, v2;
```

```
  SERIALIZABLE(Class);
public:
  C() : v1(12), v2(24) {}
};
```

Yep. But using macro is ugly and requires more typing. Instead, there is

```
template <class T> XmlIO XmlIO::operator()(const char *tag, T& var) {
 XmlIO n(*this, tag);
 Xmlize(n, var);
 return *this;
}
```

soo if you want to be aspect aware, which I understand as being able to add xmlization to existing class, simply define Xmlize *function* for it.

Xmlize itself has template

```
template <class T>
void Xmlize(XmlIO xml, T& var)
{
 var.Xmlize(xml);
}
```

(Now if somebody asks what we mean by "aggresive use of C++", here is the answer).

Note that sometimes you might want to define more funtions to xmlize single type to express the value the type really represents, e.g. there is XmlizeLang that interprets int as language indentifier (e.g. "en-us").

Quote:
About a uppforge, why waiting high demand? Won't the offer create demand? At least, this shall only encourage sharing U++ code, thus giving more examples of U++ code. Maybe just letting a space where to put code snipset would be nice.

What I really wanted to say is that U++ is not popular enough for somebody to start one. If you feel like starting such effort, go for it, it will definitely help us.

Mirek

Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 23:10:46 GMT
View Forum Message <> Reply to Message

Maybe, you miss the point, because your solution forces to modify class C with adding a method C::Xmlize(). This means modifying header file.
And I also meant I wanted to add new streams kind (like ShallowTraceOutStream, DeepTraceOutStream or whatever I can imagine, DBstream, or LuaStream).
When doing this I don't want to be intrusive and modify U++ headers, to add String::ShallowTraceOutStream(), Vector<>::ShallowTraceOutStream()...
Hence I don't need to modify U++ core with this pattern. The cost is breaking the encapsulation.
But shouldn't be too maniac about that, in this case benefits are bigger.

Subject: Re: What about LUA plugin?
Posted by qwerty on Sun, 10 Sep 2006 23:12:33 GMT
View Forum Message <> Reply to Message

uh, I almost oversee this topic .

Lua plugin? yes, I have it somewhere. the question is, if you want port it to c++. native lua supports C (*func) only, so there are some interesting projects to make it posible w/ C++, I guess, the choice is yours(www.lua.org). if there is interest to upload somewhere plugin, which I am using(personally, it's not worth of to tell a "plugin", just the slight modified/created files), drop the note.

Lua is slow?? heh heh, yes and the sun rises on the west side :>

Subject: Re: What about LUA plugin?
Posted by qwerty on Sun, 10 Sep 2006 23:21:43 GMT
View Forum Message <> Reply to Message

btw, if there will be a interest, I could try the best option with Lua and C++ phenomenon and make a "full" plugin this way. these days, I am using a Diluculum plugin, which is simply enought, but lack inheritance and some minorities...

of course, everything with(in) U++ framework

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 23:22:43 GMT
View Forum Message <> Reply to Message

thierry wrote on Sun, 10 September 2006 19:10Maybe, you miss the point, because your solution

forces to modify class C with adding a method C::Xmlize().

No, you missed it

You just have to define your Xmlize *function*. You do not need to alter header to do that.

Quote:
And I also meant I wanted to add new streams kind (like ShallowTraceOutStream, DeepTraceOutStream or whatever I can imagine, DBstream, or LuaStream).

XmlIO is really not a stream and cannot be a stream - it is in fact an interface to map hierarchy (that is what XML file is...). Well, maybe in future it would be nice to virtualize XmlIO to support other map hierarchies. At the time of creating XmlIO, it seem as far fetched idea from practical point of view.

Mirek

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 23:24:55 GMT
View Forum Message <> Reply to Message

qwerty wrote on Sun, 10 September 2006 19:12
if there is interest to upload somewhere plugin, which I am using(personally, it's not worth of to tell a "plugin", just the slight modified/created files), drop the note.

I guess, upload it here

Mirek

Subject: Re: What about LUA plugin?
Posted by thierry on Sun, 10 Sep 2006 23:39:39 GMT
View Forum Message <> Reply to Message

No no you have missed it

```
void Xmlize(XmlIO xml, C& var) {
  // Did C tell you were a friend? No? Back off then !
}
```

Defining a function overriding the template doesn't give access to C protected/private member, so you have to modify C either to make a friend void Xmlize(XmlIO xml, C& var) or to implement it as

C::Xmlize(), which I then tend to prefer.

Subject: Re: What about LUA plugin?
Posted by mirek on Sun, 10 Sep 2006 23:49:25 GMT
View Forum Message <> Reply to Message

thierry wrote on Sun, 10 September 2006 19:39No no you have missed it

```
void Xmlize(XmlIO xml, C& var) {
  // Did C tell you were a friend? No? Back off then !
}
```

Defining a function overriding the template doesn't give access to C protected/private member, so you have to modify C either to make a friend void Xmlize(XmlIO xml, C& var) or to implement it as C::Xmlize(), which I then tend to prefer.

A valid point. Usually, you can setup the value of instance based on its methods (think std::string), but you are right that this is not a rule. Anyway, once properties are not public, there is no other means to access them than method....

Mirek

Subject: Re: What about LUA plugin?
Posted by qwerty on Sun, 10 Sep 2006 23:51:46 GMT
View Forum Message <> Reply to Message

don't want to interfere, just plugin here...

- add package to project
- include <...lua/lua.hpp>

please, our guru, review it, if its ok. it works in my appz both in win, lin

PS: I will do something with c++ port to include it in; gimme some time. Diluculum needs boost library, so I'll choose something else...

File Attachments
1) lua.zip, downloaded 2035 times

Subject: Re: What about LUA plugin?
Posted by mirek on Mon, 11 Sep 2006 06:59:37 GMT

View Forum Message <> Reply to Message

Some simple example to test with?

Mirek

Subject: Re: What about LUA plugin?
Posted by qwerty on Mon, 11 Sep 2006 09:38:30 GMT
View Forum Message <> Reply to Message

```
#include <Core/Core.h>
#include <plugin/lua/lua.hpp>


CONSOLE_APP_MAIN
{
 lua_State * moon = lua_open();

 luaL_openlibs(moon);
 luaL_dostring(moon, "print(\"hi honey\")");

 lua_close(moon);

 system("pause");
}
```

maybee, it would need to change some lines in lua.hpp...

just add 'src/' before include file name in ""

Subject: Re: What about LUA plugin?
Posted by lindquist on Sun, 17 Sep 2006 15:46:14 GMT
View Forum Message <> Reply to Message

Glad to see this here. I am going to use Lua as scripting language in my app, so this will save me a little time.

Thanx

Subject: Re: What about LUA plugin?
Posted by qwerty on Sun, 15 Oct 2006 07:48:07 GMT

if u will be in need of help there, feel free to ask