Subject: Some projects are ready to become open-source
Posted by Mindtraveller on Thu, 24 Apr 2014 15:07:52 GMT
View Forum Message <> Reply to Message

Hi! It's been a while since I've posted anything to forum, but still U++ is the best thing I've used. There's a number of U++ projects I'd like to make open-source and make them as bazaar packages. But I don't really know if people really need them. Besides, some of them require additional work to become really cross-platform.
So I'll post a list of possible packages, and if someone considers them useful, please vote here.

1. First of all, I'd like to present multithreaded worker-based server class. I don't know if it duplicates any of Skylark functionality (never really used it). The main idea behind it was to make lightweight and easy to use solution for high load servers. Upon incoming http-request, a free worker thread is reused or new worker thread is automatically created. The class was made with http keep-alive support in mind.

2. Persistent storages. Rather simple wrapping templates around U++ storage classes adding functionality for accessing them in multithreaded environment with background auto-save to file system. Classes are very helpful for effective management of not-very-big amounts of data where SQL-based solutions seem too "heavy". There are plans to decrease memory usage and serialization effectiveness for these storages if it comes to it.

3. BNF (Backus-Naur form) classic implementation based on stack machine. Stable but really needs upgrade to support some jit-compiled code for internal functions. I'd be very grateful if anyone is interested. I don't use BNF package very frequently alone, instead it is widely used as a dependency of next package.

4. Industrial-grade, completely asynchronous serial communication library. Based on Apache-licensed serial library. It doesn't work on send/receive bytes level. Instead, it operates with BNF-defined protocols, parses them from file upon startup. User code operates with high-level things like input and output Value vectors. If you want some simple experiments with serial port, this is too heavy solution. But if you write big project where serial communications involved, this is what you might dream of. Very robust and well tested through years.

5. ToJSTime/FromJSTime functions for browser time representation support. Very small, doesn't worth creating package for them.

6. ConvertTextToUtf8 function. Charset detection function which makes utf-8 text from almost any encoding. Uses UCI & iconv libraries. Succesfully detects almost any encoding and converts text with at least 10 characters long (smaller text might give too big detection error).

7. U++ packaged htmlcxx library. Very lightweight but accurate HTML/CSS parsing library with LGPL license.
Added utility function SafeguardHTML (with utf8 conversion from previous function). The main idea is to create 100% safe html output from user input in any encoding.

As always, any suggestions or critics are welcome.

Subject: Re: Some projects are ready to become open-source
Posted by dolik.rce on Thu, 24 Apr 2014 16:54:22 GMT
View Forum Message <> Reply to Message

Hi Pavel,

Numbers 2 and 7 sound quite useful to me :) Not that I had any project in mind that would make use of them right now, but they sound like something one would need relatively often.

Other points of course sound interesting too, especially 3 & 4, but I personally try to stay as far from communication in programs as possible :)

Best regards,
Honza

Subject: Re: Some projects are ready to become open-source
Posted by Oblivion on Thu, 24 Apr 2014 18:24:36 GMT
View Forum Message <> Reply to Message

Hello Mindtraveller,

These projects sound very nice. I'd certainly vote for 2, 5 and 7.
Especially, a Html/CSS parser would be very useful.

Regards.

Subject: Re: Some projects are ready to become open-source
Posted by deep on Fri, 25 Apr 2014 09:13:40 GMT
View Forum Message <> Reply to Message

Hi Pavel

All projects are useful.
I vote for 1,2,3,4.

Have you used BNF for HTML parser?

Subject: Re: Some projects are ready to become open-source
Posted by forlano on Fri, 25 Apr 2014 11:30:56 GMT
View Forum Message <> Reply to Message

Hi Pavel,

I vote for number 7.

Thanks,

Luigi

Subject: Re: Some projects are ready to become open-source
Posted by mirek on Sat, 26 Apr 2014 09:42:41 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 24 April 2014 17:07
7. U++ packaged htmlcxx library. Very lightweight but accurate HTML/CSS parsing library with LGPL license.
Added utility function SafeguardHTML (with utf8 conversion from previous function). The main idea is to create 100% safe html output from user input in any encoding.

I am quite interested in this one...

Actually, I was planning to do some HTML->RichText decoder, simply because I need it for my work.

Mirek

Subject: Re: Some projects are ready to become open-source
Posted by mirek on Sat, 26 Apr 2014 09:44:25 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 24 April 2014 17:07

5. ToJSTime/FromJSTime functions for browser time representation support. Very small, doesn't worth creating package for them.

[/quote]

It is possible that this duplicates Core's:

String WwwFormat(Time tm);
bool   ScanWwwTime(const char *s, Time& tm);
Time   ScanWwwTime(const char *s);

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 21:21:58 GMT
View Forum Message <> Reply to Message

I've uploaded sources. Comments and description is added below.

File Attachments
1) source.zip, downloaded 534 times

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 21:29:15 GMT
View Forum Message <> Reply to Message

MtAlt - base package used by other packages, version 2.0. This is alternative approach to multithreading discussed here:
http://www.ultimatepp.org/forums/index.php?t=msg&goto=25 087#msg_num_7
This new version introduces bounds for callback queues, priorities and much better debugging support including runtime latency measurements with file/line indication (everything is completely wiped out release mode leaving no overhead).
Frankly speaking, it's a pity I was unable to write good enough documentation for this package, so people consider this approach too unnatural and don't use it. But proper usage of this package may eliminate many problems when you have 3-5-10 or even more threads.
Please update Bazaar/MtAlt package with this new version.

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 21:34:04 GMT
View Forum Message <> Reply to Message

BNF - special way to parse and generate raw data using some protocol. For example: http://en.wikipedia.org/wiki/Syntax_diagram#Example
Package contains classes describing BNF-articles. Each BNF-article is capable of two main functions: parsing and generation, which means converting a number of Values from/to the String. You may construct these articles using their operators, but package gives you much more powerful way. Calling CreateDefaultBNFFunctions() creates rules to construct new BNF-articles from EBNF description.
Parsing/generation process is made within primitive virtual state machine using stack and registers.
If you need custom parse/gen function working within your BNF description, you may write it and register it in runtime with bnfParseProcs.Add()/bnfGenProcs.Add() calls. There's already added bunch of functions to support binary protocols and various crc types.
It is working and it is stable. But currently I don't think that primitive stack/register virtual state machine is the best choice here. So I'd be grateful if anyone interested in optimizing the machine will do the thing. Of course, I'll be in touch and answer all the questions and/or help with upgrade to more optimized version.

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 21:44:12 GMT
View Forum Message <> Reply to Message

SerialPort - not just a simple serial port package. It is based on MtAlt and BNF.
Long story short, it loads the file with description of each serial port to be used. A multithreaded class is created for each port. It also loads a file with protocol descriptions (i.e. modbus,...).
Let's look how you request i/o with serial port:

```
 void SendReceive_
 (
  String        sendProto,
  Vector<Value>  sendArgs,
  String        receiveProto,
  dword          timeout,
  void          *custom,
  Ptr<OBJECT>    notify,
  void (OBJECT::*cb)(bool success, Vector<Value> args, void *custom),
  bool          useAcc = false,
  bool          priority = false
 )
```

Ending underscore means this is asynchronous function which may be called from any other thread ("mt safe").
sendProto and sendArgs are send data generation protocol and it's variables (string, integer, float, etc.)
receiveProto is data parsing protocol to parse incoming data. As soon as incoming bytes are parsed successfully, incoming datagram is considered succesfully received.
timeout is a receive timeout which is monitored by internal thread.
custom - is just a custom value to distinguish multiple SendReceive_() calls from each other.
notify/cb - is a CallbackQueue/CallbackThread object and it's member function which is called upon send/receive/timeout event.
useAcc - is a "caching" feature; Serial port classes keep some cache of incoming byte sequences already parsed. And if the same sequence is met, no actual parsing is made, instead returning cached result. This is extremely useful for slow processors like early celerons or arm. My tests revealed that more than 60% of incoming packages are the same and thus are cached very effectively leaving many processor resources for other tasks.
priority - is a simple true/false switch putting your call into the front of queue of into the end of it. Yes, just imagine you call SendReceive_() before previous call is still working. According to MtAlt, all subsequent calls are added to the queue (not necessarily to the tail, because we have priorities).

This package needs further discussion and examples which will follow if someone interests. Should we make new topic for it?

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 22:16:06 GMT
View Forum Message <> Reply to Message

Utils - a minified part of utility package from one of my projects. It is not currently compiled and linked cross-platform. Actually it is not intended to be standalone package, but rather a bunch of possibly useful functions using:
- ICU (http://site.icu-project.org/) a standard Linux/Unix library also present for Windows (may be even inside standard libs, I don't remember exactly).
- iconv a standard Linux/Unix library for charset conversion
- embedded htmlcxx library for html/css parsing

Interesting functions:
Encrypt/Decrypt - very simple AESStream wrapper for small strings conversion. I propose adding them to AESStream itself.
ToJSTime/FromJSTime - very simple conversion from/to javascript "integer" Time/Date representation.
ConvertTextToUtf8 - ICU's charset detector result routed to iconv library making output text with utf-8 charset. My tests introduced excellent results for big texts (about 1 KB) and rather good results for texts above 100 bytes. If text is about ~10 bytes, the charset detection generally results in wrong detected charsets.
SafeguardPlain - web-oriented function to convert text added by user to make it 100% safe before embedding it to web page.
SafeguardHTML - web-oriented function to convert html to make it 100% safe before embedding it to web page. The only way of doing it is parsing input html (with htmlcxx), then filtering it (with HTMLFilter class), and then reconstructing resulting html back again. You may of course take htmlcxx library from this package and use it the way you want (using SafeguardHTML as an example of htmlcxx usage).
ParseFormData - is a function to parse input form-data from socket (actually it is little outdated, but it is no problem rewriting it for TcpSocket). The main idea behind this function is to create streaming solution for downloading large files from Internet to the drive. So ordinary form fields are added to fields variable, and files are streamed directly to file system using file names from files variable. Very useful when working with big data in servers.

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 22:38:07 GMT
View Forum Message <> Reply to Message

PersistentMap - a bunch of templates to make U++ vectors/maps persistent.
The idea behind it is very simple:
1) Take original storage class
2) Restrict or don't restrict access to it storage itself depending on flavour.
3) Add functions for "mt safety".
4) Add background thread to write data to disk if storage is changed with some timeout (if it is changed frequently).
5) Make writes atomic and stable across situation of write failure / io failure / power failure, so that user code transparently works with stored data even after failure.

currently added is
6) Versioning support for "open" flavour

and to be done is
7) Clustered incremental disk flushes for partial and effective disk writes.

First flavour is "restricted" PersistentStorage/PersistentMap<STORAGE>. This template doesn't give access to underlying container. Instead it introduces functions like Add, AddPick, Get, GetFirst, Put, CheckAdd or GetRemoveFirst. You may be sure about it's "mt safety" and persistency.

But this is not always enough. Sometimes you need more flexibility, and here you have "open" flavour, a OpenPersistentStorage<STORAGE> class. It gives you full access to underlying storage, but you are responsible for calling EnterRead()/LeaveRead() and EnterWrite()/LeaveWriteSetChanged(bool changed).

Currently these templates are in release stage and you may use them without any doubt of losing your data. Any suggestions or proposals are welcome.

File Attachments

1) PersistentMap.h, downloaded 478 times

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Tue, 06 May 2014 23:15:46 GMT
View Forum Message <> Reply to Message

ServerWorker - a multithreaded high load server.
I'm of course aware of modern Skylark functionality created by Mirek. But there are cases when you need not a front-end server, but instead you need very effective intermediate processing server which effectively handles multiple connections simultaneously.
The base concept of this approach is a "worker". Worker is an instance executed in it's own thread which processes single user's request and finishes. In my implementation, if there's a free worker it is used for new request, or new worker/thread is added.
This approach leads to very efficient and extremely lightweight server which handles large amount of multiple connections.
Actually I think it is one of useful things among published, but it is up to you - if you consider it useful for your tasks (high load is not a kind of task you frequently meet in real life).

To demonstrate the class is really easy to use, I'll post example here.

How server with custom workers is created and used:

```
TcpSocket server;
server.Listen(port,connections);
ServerWorker::ServerThread<CommentWorker>(server);
ServerWorker::ClearWorkers_();
```

Making custom worker itself:

```
#ifndef _commentd_CommentWorker_h_
#define _commentd_CommentWorker_h_
```

```
#include <Core/Core.h>
#include "ServerWorker.h"
using namespace Upp;

class CommentWorker : public ServerWorker
{
public:
 CommentWorker()
 {
  ServerWorker::AddHandler<CommentWorker>("/login",
&CommentWorker::HandleAddLogin);
  ServerWorker::AddHandler<CommentWorker>("/comment",
&CommentWorker::HandleAddComment);
  //...
 }

 void HandleAddLogin(const String &request, HttpHeader &header, TcpSocket &socket)
 {
  VectorMap<String,String> values = ParseUrlQueryString(header.GetURI());
  String valueLogin = values.GetAdd("login");
  String valuePass = values.GetAdd("pass");
  //...
  HttpResponseKeepAlive(
   socket, false, 200, "OK", "text/html; charset=\"utf-8\"",
   HTML, NULL,
   VectorMap<String,String>()
  );
 }

 //...
};

#endif
```

That's it. You don't care about creating threads or workers. You just write handlers keeping in mind the multithreaded nature of your code. PersistentStorage classes fit here perfectly.
The current state of this class is almost beta. It is working but some non critical issues are found (i.e. longer answer timeout on Windows) as well as functionality itself is yet to be finalized.

I'd be very grateful on your suggestions and thought on this topic.

File Attachments

1) ServerWorker.h, downloaded 614 times

Subject: Re: Some projects are ready to become open-source
Posted by rxantos on Wed, 07 May 2014 03:58:25 GMT

View Forum Message <> Reply to Message

[quote title=Mindtraveller wrote on Thu, 24 April 2014 11:07]
7. U++ packaged htmlcxx library. Very lightweight but accurate HTML/CSS parsing library with LGPL license.
/quote]

Unless you are using some pre-existing code that already have this license. Why is there a need for it to be LGPL?
Why not a license compatible with BSD? (like the rest of UPP, with the exception of some in Bazaar).

http://www.ultimatepp.org/appideAbout$en-us.html

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Wed, 07 May 2014 08:15:42 GMT
View Forum Message <> Reply to Message

rxantos wrote on Wed, 07 May 2014 07:58Why is there a need for it to be LGPL?
I needed lightweight html parser library without any additional dependencies. Then I found htmlcxx which was LGPL.
If you find lightweight library with BSD license which is effective and mature enough - please make wrapper around it, and I'll replace htmlcxx with it.

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Wed, 18 Jun 2014 16:03:37 GMT
View Forum Message <> Reply to Message

OK, it's been some time since I've uploaded sources and it's actually no feedback at all.
Did someone find any of sources useful?
Is it complete mess and you just didn't want to dig into such a horrible code?
What are main problems with embedding these classes into your projects?
Are those classes badly organized and used some alien, hardly understandable concepts?
Should we just abandon them? Or any of them worth adding to bazaar?
Please give me some feedback if you spent some time learning the code posted.
Thank you in forward.

Subject: Re: Some projects are ready to become open-source
Posted by deep on Sat, 01 Nov 2014 02:27:47 GMT
View Forum Message <> Reply to Message

Hi Pavel,

I want to use Serial Port code for handling ModBus protocol.

Any small example will be useful to me. Using BNF and MtAlt.

Mindtraveller wrote on Tue, 06 May 2014 21:44SerialPort - not just a simple serial port package. It is based on MtAlt and BNF.
....
This package needs further discussion and examples which will follow if someone interests.
Should we make new topic for it?

Subject: Re: Some projects are ready to become open-source
Posted by jerson on Sat, 01 Nov 2014 05:24:16 GMT
View Forum Message <> Reply to Message

Yes I second that request.

I had to implement modbus protocol, but since I could not understand this, I had to roll my own -newbie- code that works.

I'd appreciate an example for a single command of modbus and how it could be implemented using this package.

Regards
Jerson

Subject: Re: Some projects are ready to become open-source
Posted by Mindtraveller on Mon, 10 Nov 2014 14:26:57 GMT
View Forum Message <> Reply to Message

Hi, I've attached simple SerialPort-based project to this message. It is really very minimalistic and well-documented.
Files 'ports' and 'proto' must be placed into executable's directory.
'ports' file contains ports used. Ports are referenced by 'name' attribute.
'proto' file contains descriptions of protocols used (currently only 'modbus' is used).

File Attachments
1) serialTest.zip, downloaded 524 times

Subject: Re: Some projects are ready to become open-source
Posted by deep on Tue, 11 Nov 2014 02:06:36 GMT
View Forum Message <> Reply to Message

Hi Mindtraveller,

Thank you for posting example.
Will check and get back.