Subject: FTP class and reference example for U++
Posted by Oblivion on Sun, 27 Apr 2014 16:11:11 GMT
View Forum Message <> Reply to Message

Hello,

Ftp class is a U++ client side implementation of the File Transfer Protocol (FTP) as specified in RFC 959, with several advanced capabilities:

- Time-constrained blocking, and non blocking operation modes.
- Multithreaded file transfers, using worker threads.
- IPv6 connections and NATs, as specified in RFC 2428.
- Ftp over TLS/SSL (FTPS), as specified in RFC 2228.
- Feature negotiation mechanism as specifien in RFC 2389.
- Parsing of UNIX and DOS style directory listings.
- Extending the existing functionality of Ftp class.
- Transfer restart/resume mechanism, as specified in RFC 3959.
- UTF-8 encoded path names.

Reference examples provided with the package:

- FtpGet: Demonstrates a basic FTP file download in blocking mode.
- FtpGetNB: Demonstrates a basic FTP file download in non-blocking mode.
- FtpGetMT: Demonstrates a basic Ftp file download, using worker threads.
- FtpMultiGetMT: Demonstrates FTP dir listing and concurrent file transfers, using worker threads.
- FtpGUI: Demonstrates a basic FTP browser with GUI (with upload, download, mkdir, rename, delete commands).
- FtpOverTLS: Demonstrates the secure connection capability of FTP package in blocking mode.
- FtpQueryFeatures: Demonstrates the feature query mechanism, as defined in RFC 2389
- FtpRawCommand: Demonstrates FTP raw command execution in blocking mode.

Latest code is located at: https://github.com/ismail-yilmaz/upp-components/tree/master/Core/FTP
Reference examples can be found at:
https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples
Older version (1.2) can be found at:
https://github.com/ismail-yilmaz/upp-components/tree/master/ Attic/FTP

A video demonstration of Ftp browser: (Note that the example browser is still under development.)

https://vimeo.com/214530673

Suggestions, bug reports, patches, criticism are always welcome.

Regards,
Oblivion

File Attachments

1) FtpPackageWithExamples.zip, downloaded 454 times

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Tue, 06 May 2014 23:28:01 GMT
View Forum Message <> Reply to Message

Hello guys,

Today I publish the first public beta of the FTP package (Link to the updated versions of the package can be found in the first post).

What is missing: Aside from the todo list in the first post, currently only GetReplyAsXml() method is missing,. but it will be added within this week. Also, I removed most of the internal error messages, I'll add them later.
I also wrote a FtpBrowser with a non-blocking GUI (not multithreaded), which will demonstrate how easy it is to set up a non-blcoking gui with Ftp and on U++.. It will be added as a reference example.
But not uploaded this time (code needs cleaning). Ftp class is also a demonstartion of TcpSocket based Server and Client sockets, since it actively uses both type of sockets.

And of course there will be bugs. I am sure because Ftp protocol is, by itself, a total mess. :)

So, suggestions, bug reports, patches are always welcome.

Ftp class is tested under Linux 3.12+ and Windows (XP/7).

Cheers!

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Thu, 08 May 2014 09:52:05 GMT
View Forum Message <> Reply to Message

Update: NOOP command is added to the Ftp class. (See the first post for package), .

Regards.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Tue, 13 May 2014 00:03:53 GMT
View Forum Message <> Reply to Message

Hello guys,

I updated the Ftp class and reference examples package (see the first post for package).
Reference examples package now contains a very simple (for demonstration only) Ftp Browser with non-blocking GUI.
This version of the example allows only one download at a time, but a multithreaded example which will allow multiple downloads, will follow. (It will be very similar to the U++ Reference/GuiMT example),
FtpBrowser utilizes two ftp objects (a browser and a worker). This makes possible to navigate while download. Also it demonstrates how to use FtpDirEntry class in conjunction with the FileList class.
Here is the screenshot of the ftp browser example.

Bug reports, suggestions, requests, etc. are always welcome.

Regards.

File Attachments
1) FtpBrowserExample_Screenshot.jpg, downloaded 1777 times

Subject: Re: FTP class and reference example for U++
Posted by mirek on Wed, 14 May 2014 07:25:36 GMT
View Forum Message <> Reply to Message

There already exists plugin/ftp. Could you please summarize advantages of your new implementation (before I will dwell into it myself...)?

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Wed, 14 May 2014 11:43:49 GMT
View Forum Message <> Reply to Message

Hello Mirek,

While it is far from perfect, and I'm not sure if the below mentioned features count as advantages, aside from the basic operations, but here they are:

1) Ftp class is completely based on native Upp classes. It is derived from TcpSocket. Therefore supports its features (except asynchronous connections, due to FTP protocols synchronic nature).
This makes bug tracking and understanding the source code easy (also ftp class source code is

around 600 lines, and meant to be clean and concise). No external lib, no C code, no need for external memory management, etc..
2) It comes with a directory entry and file information parser (FtpDirEntry class) which makes parsing (both DOS and UNIX based directory listing) as simple as it can be (of course getting the raw Listing strings is also possible (see CLI based ftp example).
3) It is designed the U++ GUI in mind; For example, FtpDirEntry class perfectly fits with FileList class (see the supplied ftp browser example). and achieving a non-blocking gui is very easy.
4) Ftp class does work on U++ streams (any kind), not simply strings (so it can take advantage of FileIn or FileOut streams, without any other effort, which means reduced code).
5) Ftp class api can be extended via SendCommand method (see CLI based ftp example) by the user/developer (there are many useful FTP extension commands out there, added later).
6) Ftp class is not just based on RFC 959. It also takes into account some practical deviations (D. J. Bernstein's recommendations: http://cr.yp.to/ftp.html) from the original draft.
7) Proper support for aborting transfers.
8) Ftp class will support IPV6 (actually It already does in development branch, and I will merge it soon).
9) Ftp class will support proxies.
10) I am also planning to add SSL/TLS support.
11) It is documented. (API document is complete, and I am currently writing a document on its mode of operations).
12) It is actively developed.

I believe the code is in good shape (after all, it is in beta state), but there is always plenty of room for improvement, and bugs that I don't know of yet.

Regards.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Wed, 24 Dec 2014 11:06:28 GMT
View Forum Message <> Reply to Message

Hello guys,

Ftp class is updated (rewritten, actually). It now takes advantage of EPRT and EPSV commands, which means it has from now on IPv6 support.

See first post for details and package.

Regards,
Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Thu, 25 Dec 2014 17:03:52 GMT
View Forum Message <> Reply to Message

Hello,

Two convenience functions are added to the Ftp package: FtpGet and FtpPut:

int FtpGet(const String& path, Stream& out, const String& host, int port = 21, const String& user = Null, const String& pass = Null,
Gate2<int64, int64> progress = false, Callback whenwait = CNULL, int type = Ftp::BINARY, int mode = Ftp::PASSIVE);

int FtpPut(Stream& in, const String& path, const String& host, int port = 21, const String& user = Null, const String& pass = Null,
Gate2<int64, int64> progress = false, Callback whenwait = CNULL, int type = Ftp::BINARY, int mode = Ftp::PASSIVE);

These functions can simplify Ftp download and upload operations, while retaining the flexibility of the Ftp class. Ftp browser example is also updated to reflect this addition.

Regards,
Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Wed, 07 Jan 2015 19:52:00 GMT
View Forum Message <> Reply to Message

Hello,

Ftp class gained basic FTPS (Ftp over SSL, to be specific) support. From now on Ftp class can connect to secure Ftp servers, using "explicit mode" negotiation. See RFC 2228 for details.

There are some limitations though:

1) Currently only SSL is supported. TLS is not yet supported.
2) FTPS can be only activated per ftp session. Namely, it must be enabled or disabled prior to an Ftp::Connect() call.
3) Active connection mode and FTPS are mutually exclusive. When FTPS is enabled, the ftp data retrieval methods (such as Ftp::ListDir(), Ftp::Put() or Ftp::Get(), or convenience funtions FtpPut() and FtpGet()) will fail in active connection (PORT, EPRT) mode.

Bu reports, criticism, patches, etc. are welcome.

Regards,
Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Tue, 20 Jan 2015 00:17:57 GMT
View Forum Message <> Reply to Message

Hello,

It seems that U++ Core/SSL package already supports TLS connections (see: http://www.ultimatepp.org/forums/index.php?t=msg&th=9090 &goto=43822&#msg_43822 and https://www.openssl.org/docs/ssl/SSL_CTX_new.html).
So I added AUTH TLS support to the Ftp class.
I also tested this new feature with a secure ftp server which requires TLS and it apparently works well.

Ftp class will first explicity request a TLS mode and in case of rejection it will try SSL mode.
See first post for the details and the updated package.

Suggestions, bug reports, patches, criticism are always welcome.

Regards,
Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by unodgs on Tue, 07 Jul 2015 17:40:54 GMT
View Forum Message <> Reply to Message

Hi Oblivion!

Great package. It works well on windows as well but do you plan to add SFTP (SSH File Transfer Protocol)?
PS: I would be grateful for adding upload in your example.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Wed, 08 Jul 2015 06:50:38 GMT
View Forum Message <> Reply to Message

Hello Daniel!

Glad to hear that someone found it useful.

Sure. The example is rudimentary.
I will add upload/delete/rename/create etc. options to the example. And turn it into a usable FTP browser in the following days.

SFTP, on the other hand, needs some think-through. I had thought of adding a SSH client library

to U++ last year, but I did not have time to implement it then. (it is a somewhat complex protocol, and I don't want to end up creating something half-baked.)
IMO the best and easiest option is to create a U++ wrapper for libssh2 which is AFAIK cross-platform.
So, I will have spare time in the following days and I can start implementing a SFTP client utilizing all the U++ goodies and having usual U++ type API, based on libssh2.
In the meantime maybe I will also resurrect my U++ SSH client library project (But I'm not making any promises on this. :)).

What do you think?

Regards,

Oblivion

Subject: Re: FTP class and reference example for U++
Posted by unodgs on Wed, 08 Jul 2015 08:20:03 GMT
View Forum Message <> Reply to Message

For me sounds like a great news. Thank you Oblivion in advance!
PS: libssh2 looks like a good choice (pure C, easy to wrap up)

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Sat, 18 Mar 2017 22:28:49 GMT
View Forum Message <> Reply to Message

Hello guys,

It's time for some spring cleaning. :)

Ftp class is updated.

From now on it requires at least C++11.

And GetReplyAsXml() method is added. This method simplifies the parsing of FTP protocol messages and internal messages, using simple XML tags.

FtpExample is also updated so that it can work. I know it is a crude example but I'm in the process of writing a better version of it with multithreading in mind.
It'll be replaced asap.

As usual, you can find the updated package in the first post of this topic.

Regards,

Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Sun, 09 Apr 2017 00:32:52 GMT
View Forum Message <> Reply to Message

Hello,

I'll deliver on my promise and upload the new ftp browser example, along with the updated FTP package within next week.
As you can see from the below screenshot, the new browser example is quite capable.
(The funny thing is, it contains less than 400 LoC (subject to change, but not much) and most of the code is about GUI creation and management.

Yet it supports and demonstrates almost every aspect of the FTP package, including multithreading. :)

Yep, FTP package finally gained multithreaded file transfer support. 8)

I have to say that it is very simple and easy to use. In fact the multithreaded DL/UL functions (FtpAsyncGet and FtpAsyncPut) are almost identical to their single-threaded counterparts, and allow the same flexibility and integration with the Upp's core and Ctrls!

Therefore, the upcoming release will mark the version 1.0 :)

Cheers!

Oblivion

File Attachments
1) FtpBrowser.jpg, downloaded 1389 times

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Wed, 12 Apr 2017 20:34:01 GMT
View Forum Message <> Reply to Message

Hello guys,

Today marks a new milestone for FTP package. The new version (1.1) of FTP package is finally

here.
After several iterations, I believe the package has become stable.
This version brings multithreaded file transfers to the ftp class, via two convenience functions: FtpAsyncGet() and FtpAsyncPut():

```
int FtpAsyncGet(const String& remote_file, const String& local_file, const String& host, int port = 21, const String& user = Null,
	const String& pass = Null, Event<int, int, String, int64, int64> progress = CNULL, bool active = false, bool ssl = false,
	bool ascii = false);

int FtpAsyncPut(const String& local_file, const String& remote_file, const String& host, int port = 21, const String& user = Null,
	const String& pass = Null, Event<int, int, String, int64, int64> progress = CNULL, bool active = false, bool ssl = false,
	bool ascii = false);
```

As you can see the syntax is very similar to the single-threaded FtpGet() and FtpPut() functions. Except these async functions do concurrent upload/download while retaining the flexibility of the Ftp class.
Moreover, these functions are implemented in a general way that, I believe, they can serve also as a reference implementation. While the Ftp class itself is still single threaded.

Also, in this version, the FEAT command is implemented. It is now possible to query the capabilities of ftp servers easily.

One more addition to the package is a multithreaded simple FTP browser example.
Example browser can use secure connection (FTPS), download, upload, manipulate files, and manipulate directories, show file information etc.

First on my todo list is to implement REST command to allow restart/resume of interrupted transfers.

And Mirek, I know that you are a very busy person, but you'd asked me about the advantages of my ftp implementation over the current one in Core/Ftp.
I now believe FTP package is eligible to Bazaar, considering that it has IPV6, FTPS, multithreading support, better integration with U++ core and UI classes, a clean code-base, active development, is well documented, and easily extendible (using SendCommand() method), and has an example browser built around it. It makes a good alternative. What do you think?

As usual, you can find the development log and updated package in the first post of this topic.

I tested FTP package with several public FTP servers, and private servers I set up.
I can recommend,

ftp.uni-erlangen.de, User: Anonymous, Password: Anonymous.
test.rebex.net, User: demo, Password: password. Allows FTPS
demo.wftpserver.com, User: demo-user, Password: demo-user. Allows FTPS and temporary uploads.

Cheers!

Oblivion

Subject: Re: FTP class and reference example for U++
Posted by Klugier on Wed, 12 Apr 2017 21:05:33 GMT
View Forum Message <> Reply to Message

Hello,

I am not tested this package, but i would like to notice small issue with the API design. The problem is that you need to pass too many arguments to the single function, if you want to change single argument like ascii. Let's take a look at below function:

```
int FtpAsyncGet(const String& remote_file, const String& local_file, const String& host, int port = 21, const String& user = Null,
	const String& pass = Null, Event<int, int, String, int64, int64> progress = CNULL, bool active = false, bool ssl = false,
	bool ascii = false);
```

This is the poor design, because the call will look like this:

```
FtpAsyncGet("remote.txt", "local.txt", "locaclhost, 21, Null, Null, CNull, false, false, true);
```

Looks terrible. Isn't it? In my opinion the good API has got maximum three parameters - otherwise it becomes hard to use by the client.

We can replace following code with the object approach (This is the example and I highly recommended to over thing this design):

```
int FtpAsyncGet(const FtpAssyncRequestConfig& config);

FtpAsyncGet(FtpAssyncRequestConfig("remote.txt", "local.txt", "locaclhost.txt").EnableAscii());
```

```
class FtpAssyncRequestConfig final
{
public:
   FtpAssyncRequestConfig(const String& remote_file, const String& local_file, const String&
host)
      : // initialization ...
   {}

   // Get, Set, Enable interface...

private:
   // private members
};
```

Much more easier to set the n-th parameter, but require additional work for the library provider.

_____
You could replace NAMESPACE_UPP with namespace Upp { and END_NAMESPACE_UPP with }. The NAMESPACE_UPP approach was deprecated since last release.

_____
The next problem I see in the code is the ordering problem - you should put public class interface at the top then private things (Ftp and DirEntry classes). This is the general rule mentioned in following documentation topic.

_____
This code review aims to improve the code quality.

Sincerely,
Klugier

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Thu, 13 Apr 2017 06:17:33 GMT
View Forum Message <> Reply to Message

Hello Klugier,

Thank you very much for the feedback.

Quote:You could replace NAMESPACE_UPP with namespace Upp { and END_NAMESPACE_UPP with }. The NAMESPACE_UPP approach was deprecated since last release.

It seems that I missed the announcement. Will change this asap.

Quote:The next problem I see in the code is the ordering problem - you should put public class interface at the top then private things (Ftp and DirEntry classes). This is the general rule mentioned in following documentation topic.

I tend to follow the coding style of U++. It improves the readability a lot. But is this ordering mandatory? I mean even U++ has classes with members declared in "reverse" order.
In fact I used to declare class members in traditional order (public, protected, private) in the past, but changed my habit after I got used to U++ code base. :) Nevertheless, no problem.
it will only take several seconds to reverse the order. YEt, imho, this is more a matter of aesthetic taste.

Quote:
We can replace following code with the object approach

Well, yes. In fact, I initially designed it with objective approach. Bu then decided otherwise.
Not that it wasn't useful. I had simply felt that it was bloating the code (this was maybe a side effect of being a former C programmer :)).

But what you propose is reasonable, actually.
I can depreceate the current convenience functions in favour of an objective approach.
Given the flexibility of FTP class and package, it will be very easy. FtpAsyncPut() and FtpAsyncGet() functions call, under the hood, FtpAsyncIO which has one more parameter to determine the direction of transfer.
An Ftp::Request object can take care of both synchronous and asynchronous conveninence functions' parameter list, without giving any headache, acting as an intermediate layer.

FtpGet(Ftp::Request(local_file, remote_file, localhost).SSL().Passive(), progress, whenwait)

FtpAsyncGet(Ftp::Request(local_file, remote_file, localhost).SSL().Active(), progress)

Also, I can change the Event<int, int, String, int64, int64>, which, frankly, I am not happy with, into an Event<Ftp::Result> (which can be returned by also (FtpGet and FtpPut):

```
Ftp:: Result {

public:
	int    GetErrorCode() const;
	String GetErrorDesc() const;
	int64  GetTotal() const;
	int64  GetDone() const;
	// Async only.
```

```
        int    GetId() const;

        bool   IsSuccess() const;
        bool   IsFailed() const;
        // Async only.
        bool   InProgress() const;

        Result() {}

private:
// members...
};

//
// If we return Ftp::Result from FtpGet()

 if(FtpGet(...).IsSuccess())
   //....
 else
   //....

 // OR

  Event<Ftp::Result> r;
  FtpAsyncGet(..., r, ...);

  // In somewhere else, probably in the main thread:

void Foo::TransferProgress(Ftp::Result r) {

// First take care of UI thread synchronization, using whatever necessary, e.g PostCallback.
// then simply...

   if(r.InProgress())
     some_ctrl.Text(r.GetTotal(), r.GetDone());
   else
   if(r.IsSuccess())
     some_ctrl.Text("Done.");
   else
     some_ctrl.Text("Failed".);

}
```

What do you think?

Regards,

Oblivion

Subject: Re: FTP class and reference example for U++
Posted by cbpporter on Thu, 13 Apr 2017 10:31:25 GMT
View Forum Message <> Reply to Message

Does it work for HTTP(S) file gets?

I need to implement a large file downloader with pause and resume...

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Thu, 13 Apr 2017 11:55:12 GMT
View Forum Message <> Reply to Message

Hello cbpporter,

cbpporter wrote on Thu, 13 April 2017 13:31Does it work for HTTP(S) file gets?

I need to implement a large file downloader with pause and resume...

You mean Ftp over Http? Unfortunately, no. It's out of FTP package's scope.
But you can already download large files with it. Resume and pause is not implemented (yet).

Regards,

Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Fri, 14 Apr 2017 21:45:14 GMT
View Forum Message <> Reply to Message

Hello,

As Klugier suggested above, I've re-written the convenience functions, using a more object-oriented approach.
As a result, Ftp convenience functions, both single-threaded and multithreaded variants, are

changed, and much less cluttered and more extendible now.

```
// Single-threaded.
Ftp::Result FtpGet(const Ftp::Request& request, Gate<int64, int64> progress = false, Event<>
whenwait = CNULL);
Ftp::Result FtpPut(const Ftp::Request& request, Gate<int64, int64> progress = false, Event<>
whenwait = CNULL);

// Multi-threaded.
int FtpAsyncGet(Ftp::Request& request, Event<Ftp::Result> progress = CNULL);
int FtpAsyncPut(Ftp::Request& request, Event<Ftp::Result> progress = CNULL);
```

You can find the updated package in the first post of this topic.

FtpBrowser example and FTP package documentation are also updated to reflect the changes.

Regards.

Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Klugier on Sat, 15 Apr 2017 17:23:40 GMT
View Forum Message <> Reply to Message

Hello,

It is good to hear you introduced my proposals :)

_____

I see one minor things to improvement - you could change the parameter name from whenwait to whenWait it will be more readable.

```
DirEntry&       User(const String& u)                   { vm(USER) = u; return *this; }
```

Should it be SetUser(const String& u)? Moreover the u parameter is misleading for the API consumer it should be explicitly called user.

_____

You should also consider moving your enums to c++ style:

```
 enum Style      { UNIX, DOS };

enum class Type { UNIX, DOS }; // You could consider chaining the name to Type

// The user can refere:
DirEntry::Type::UNIX

instead of:
DirEntry::UNIX
```

Do we rely need this friendship declaration here?

```
        ValueMap        vm;
        Bits            owner, group, other;
        friend bool     ParseFtpDirEntry(const String& in, Vector<Ftp::DirEntry>& out);
```

Why not just create

const ValueMap& GetVm(); // Read only access - i believe ParseFtpDirEntry doesn't set anything?

IMO the variable name "vm" is also misleading. I would call it dirInfo or something more verbose. The same improvment can be used in Result class - just call it resultInfo.

(Cosmetic - tiny) The class beginning has got problem with readability. It could be:

```
class Ftp : private NoCopy {
public: // Just remove extra empty line.
 class Request;
 class Result; // Do we need forward declarations in class? I believe when we remove friendship we will not ;)

        class DirEntry final : Moveable<Ftp::DirEntry> {
```

Sincerely,
Klugier

Subject: Re: FTP class and reference example for U++
Posted by Klugier on Sat, 15 Apr 2017 17:38:58 GMT

Hello,

In my opinion handling multi-threading API using #ifdef in header is not good for the end user. Because, he/seh is obligated to use #ifdef in theire code. Personally, i would hide it inside the interface and use One container (Movable will not work for this class).

```
class Result // Interface - all methods are abstract
{
public:
  // The same as previous

  virtual bool IsAsync() = 0;

  // Mt methods are presented without #ifdef guard
};

class RegularResult : public Result // The name could be different
{
public:
   bool IsAsync() override { return false; }

   // Mt methods returning falses, -1 etc.
};

class AsyncResult : public Result
{
public:
   bool IsAsync() override { return true; }

   // Return correct values for MT
};

One<Result> result(new AsyncResult());
```

What do you think?

Sincerely,
Klugier

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Sun, 16 Apr 2017 16:28:37 GMT

Quote:Hello,

In my opinion handling multi-threading API using #ifdef in header is not good for the end user. Because, he/seh is obligated to use #ifdef in theire code. Personally, i would hide it inside the interface and use One container (Movable will not work for this class).

```
class Result // Interface - all methods are abstract
{
public:
  // The same as previous

  virtual bool IsAsync() = 0;

  // Mt methods are presented without #ifdef guard
};

class RegularResult : public Result // The name could be different
{
public:
   bool IsAsync() override { return false; }

   // Mt methods returning falses, -1 etc.
};

class AsyncResult : public Result
{
public:
   bool IsAsync() override { return true; }

   // Return correct values for MT
};

One<Result> result(new AsyncResult());
```

What do you think?

Sincerely,
Klugier

Hello Klugier,

Thank you very much for your constructive criticism and suggestions.

I believe we might have a simpler solution. :)

FtpAsyncGet() and FtpAsyncPut() functions call FtpAsyncIO().It is possible to return en error code if anything goes wrong.
So we can simply define a function, say, IsMultitihreaded(), to check the U++ multi-threading infrastructure automatically, in Ftp.cpp:

```
static inline bool IsMultithreaded()
{
#ifdef flagMT
 return true;
#else
 return false;
#endif
}
```

Then we can call it in FtpAsyncIO:

```
Ftp::Result FtpAsyncIO(Ftp::Request request, Event<Ftp::Result> progress, bool put)
{
  // Check if U++ MT is enabled.
 Ftp::Result r;
 try {
  if(!IsMultithreaded())
   throw Exc("Multithreading is not enabled.");

// ...

 }
 catch(Exc& e) {
  // Couldn'r create or run the worker thread.
  r.info("state") = "failed";
  r.info("rc")    = -1;
  r.info("msg")   = e;
  LLOG("-- FTP worker failed. Reason: " << e);
 }
 return r;
```

In this way we can remove other ifdefs, and user won't be confused.
(Multithreading as a requirement is already mentioned in API docs for each method and function.)

Note that I also changed the return value from int to Ftp::Result. (Now they are similar to single-threaded variants)

Calling an async function such as Result::InProgress() now won't do harm, it wil silently return false.

AS for your other suggestions:

ParseFtpDirEntry() actually modifies the ValueMap. It parses the string into key-value pairs. However, now I put the actual parser code into DirEntry::Parser() method, and got rid of the friend declaration.

Same thing goes for the Request class. So, the ugly forward declarations are removed.

I updated the package to reflect the changes...

Regards,

Oblivion

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Sun, 23 Apr 2017 21:41:14 GMT
View Forum Message <> Reply to Message

Hello,

Another round of updates.
This time FtpBrowser example has received updates too. But first things first:

FTP package:

multithreaded functions are further improved.

- Available worker slots are limited to 256 ftp worker thread per process.

- It is now possible to pre-set the thread priority of each worker.

- It is now possible to pass a user data (a U++ Value), using Request::UserData() and Result::GetUserData();

- Worker abort mechanism is both simplified and improved. Vector is ditched in favor of a static Bits instance.
  This reduces both memory consumption and mutex lock/unlock times. Since, previously we needed to iterate over the vector to check if a worker was registered to be aborted.
  This was time consuming. Now we simply set a bit on or off. Since the worker counter resets at 65536 and max. worker slots are limeted to 256, it should be pretty safe.
  See the FtpAsyncIO(), Ftp::AbortWorker() and Ftp::IsWorkerAborted() for details. (I'm open to suggestions, if you have better ideas.)

- Made room for NetworkProxy support. I will add NetworkProxy support (HTTP/SOCKS4/SOCKS5) via a preprocessor directive (USEPROXY flag), since NetworkProxy package is currently not in bazaar.

FtpBrowser example:

- It is now possible to multiselect the files to be downloaded or uploaded.

- FtpBrowser no longer asks for download location for each download. A default download location can be selected. And it can be changed via environment settings.

- FtpBrowser gained file editing support. While it is still a barebone example with ability to edit only one file at a time, it will change in the near future.
  (Mirek had suggested this when we were discussing SSH package last year on "Developer's Corner", which is on it's way, by the way.)
  Of course this is far from what he suggested, but it is operational in it's current state anyway. :)

That's it for now. Two features (thread priority settings for ftp worker threads, and a ftp text editor for FtpBrowser example) are this weeks highlights.

You can find the updated FTP package in the first post of this topic.

Cheers!

Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Mon, 24 Apr 2017 17:01:15 GMT

Hello

I've made a video, demonstrating the basic capabilities (concurrent downloads of large files [three 4.5 GB files], remote file editing, browsing, etc.) of FTP package and browser.

Note that the example browser is still under development.)

Just select 1080p (HD)

https://vimeo.com/214530673

Cheers!

Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Klugier on Mon, 24 Apr 2017 20:12:27 GMT

Hello,

For me it is a great news that somebody published video related to U++. You could consider posting URL to our site in the movie description ;)

Backing to FTP - did you consider moving to enum class (c++11 feature) instead of old-c like enum. I think the whole API will gained on this change. For example:

```
class DirEntry {
// ...
enum Style      { UNIX, DOS }; // In my opinion "Type" world is a better alternative in this situation...
enum class Type { UNIX, DOS };
```

Usage:

```
if (dirType == DirEntry::Type::UNIX) {
  // .. do something
}
```

instead of

```
if (dirType == DirEntry::UNIX) {
```

```
   // ...
}
```

Much more readable. Isn't it?

Let me ask you one question - do you have GitHub repository for that project?

Sincerely,
Klugier

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Tue, 25 Apr 2017 06:37:00 GMT
View Forum Message <> Reply to Message

Quote:For me it is a great news that somebody published video related to U++. You could consider posting URL to our site in the movie description Wink

Done. Credits go where it's due. :)

Quote:Backing to FTP - did you consider moving to enum class (c++11 feature) instead of old-c like enum. I think the whole API will gained on this change. For example:

Ok, I'll change this, it won't hurt much, all it requires is some minor adjustments.

Quote:// In my opinion "Type" world is a better alternative in this situation...

Well this is more about semantics. According to Wiktionary and Oxford Dictionary, a style is a manner of doing, or presenting things, which is what directory listing in FTP is all about.
It does not necessarily say anything about the underlying file system or structure (FTP servers can and sometimes do have UNIX style dir listing, when they actually use Windows/DOS based file structure (and there are even newer and more detailed listing styles (See: https://tools.ietf.org/html/rfc3659) which I'm planning to support in next versions. That's why I refactored the Ftp::DirEntry class, using a ValueMap. It gives us a tremendous flexibility to support almost any dir listing syle used in ftp servers. Parsing of dir listings in ftp is a PITA, with no reliable or standard format around, by the way :))
It is about the way of presenting directory entries (they are same type of objects in an ftp context: both are directory listings). So, I'd prefer it stay as it is.

Quote:Let me ask you one question - do you have GitHub repository for that project?

I do use and prefer git and github for my C/Python/Perl code, but I prefer using SVN for my U++ related projects, for it is supported by TheIDE. :)
In 2015 I'd created a sourceforge SVN repo, named Ultimate++ Components (of course, with a disclaimer that the repository and the code it contains is not endorsed by
U++ in any way, and that it is simply an unofficial project to externally support the U++ framework.)
Initially it saw some activity, but I had to focus on my other projects and my job, so the repo lied dormant for a long time.
Not anymore :) I revived it some time ago, and restructuring it now.
I'm uploading the development versions of the components I develop and use, with BSD license.

https://sourceforge.net/projects/ultimatecomponents/

Regards,

Oblivion

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Tue, 13 Jun 2017 22:06:15 GMT
View Forum Message <> Reply to Message

Hello,

FTP package is updated.
With this new version, Ftp class finally gained transfer restart (resume) capability for both (binary) uploads and downloads. :)

The update is also reflected in the ftp browser example.
FTP browser example can resume/restart failed or aborted transfers (if the restart feature is supported by the connected server, of course.)
Also the ftp browser now allows drag and drop (for uploads).

Also, you'll notice two nested packages within the FtpBrowser example: BrowserCtrl and TrackerCtrl.
These widgets, which I'm preparing to publish as seperate packages, are resuable ctrls.

BrowserCtrl is very similar to FileSel, but is embeddable and provides a simple api for a flexible, file system transparent file browser.
It is under heavy development.

TrackerCtrl is a transfer tracker. It has a generic api which can be used for tracking different type of file transfers (local file copy, SFTP, FTP, WEBDAV etc.) at once.

It is under heavy development too.

Which brings us to the next step: SSH (SFTP) package. Finally I have time to finish the technical preview version of SSH package.
I'll publish it within couple of weeks. And turn the FtpBrowser into a profile-based multibrowser capable of handling Local files, FTP, SFTP, WEBDAV (my next goal), etc.

Any suggestions, criticism, bug reports are always welcome.

Updated package can be found in the first post of this topic.

Tested on Arch Linux (kernel: 4 11.3) and Windows 7/10

Cheers!

Oblivion.

File Attachments
1) FtpBrowser.png, downloaded 835 times

Subject: Re: FTP class and reference example for U++
Posted by Tom1 on Fri, 16 Jun 2017 08:23:15 GMT
View Forum Message <> Reply to Message

Hi Oblivion,

This looks nice!

I'm looking forward to see the SFTP part... :)

Best regards,

Tom

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Thu, 13 Jul 2017 15:47:52 GMT
View Forum Message <> Reply to Message

Hello,

FTP package is updated to version 1.2.1.
This version fixes a bug in the restart/resume mechanism.

FtpBrowser is not updated yet.

You can find the package in the first post of this topic.

Best regards,
Oblivion.

Subject: Re: FTP class and reference example for U++
Posted by Oblivion on Sun, 01 Apr 2018 22:17:07 GMT
View Forum Message <> Reply to Message

Hello,

It's been a long while since the last update.
A new and major release of FTP class is finally here: Version 2.0.0

New version is far superior to the older (v 1.2) version of FTP package. Basically it is simpler yet more powerful.
API-wise, although I've tried hard to keep the old interface intact, and partially succeded in this, some breakages were necessary.
Yet I will also maintain the old code for some time.

I had to redesign the Ftp class to let it take advantage of the non-blocking interface I used in SSH pacakge. Now they are almost identical. :)

Here are the major changes:

MAJOR RELEASE: Ftp package, version 2.0.0
Necessary API breakage.
Ftp is redesigned, using an effective, and simple queue-model similar to the SSH package's.
Old multithreading code completely ditched in favor of the new AsyncWork based ftp workers.
This resulted in noticable performance gain, and simplicity.
Non-blocking mode is finally implemented. Ftp class can do non-blocking calls easily.
FTPS mode improved. It is now possible to inspect the SSL information after handshake.
Support for user-defined consumer functions (for incoming data) is added.
A simple ftp URL scheme is implemented. This scheme replaces (partially) the old Ftp::Request class.
UTF-8 path name encoding support is added.
Optional Connect/Login() methods pair added. It is now possible to execute commands before login.
Ftp::DirEntry now uses pcre to parse listing strings.
Logging is improved.
Reference examples are added to the package.

Reference examples provided with the package:

- FtpGet: Demonstrates a basic FTP file download in blocking mode.
- FtpGetNB: Demonstrates a basic FTP file download in non-blocking mode.
- FtpGetMT: Demonstrates a basic Ftp file download, using worker threads.
- FtpMultiGetMT: Demonstrates FTP dir listing and concurrent file transfers, using worker threads.
- FtpGUI: Demonstrates a basic FTP browser with GUI (with upload, download, mkdir, rename, delete commands).
- FtpOverTLS: Demonstrates the secure connection capability of FTP package in blocking mode.
- FtpQueryFeatures: Demonstrates the feature query mechanism, as defined in RFC 2389
- FtpRawCommand: Demonstrates FTP raw command execution in blocking mode.

As usual, the updated package can be found in the first message of this topic.
And you can also grab it from:

Latest code is located at: https://github.com/ismail-yilmaz/upp-components/tree/master/Core/FTP
Reference examples can be found at:
https://github.com/ismail-yilmaz/upp-components/tree/master/ Examples
Older version (1.2) can be found at:
https://github.com/ismail-yilmaz/upp-components/tree/master/ Attic/FTP

A screenshot of the simple FtpGUI example (GUI is inspired by the qt's example):

Suggestions, bug reports, patches, etc. are appreciated.

Best regards,
Oblivion

File Attachments

1) FtpGUI.png, downloaded 753 times