Subject: Socket - send multiple lines
Posted by nlneilson on Tue, 09 Nov 2010 22:25:47 GMT
View Forum Message <> Reply to Message

I have been using a socket client that works OK but just sends/receives a line at a time.

I would like to send about 50 lines at a time, ~2KB
I suppose this would be the same as sending a text file.

```
#ifndef _ConnSock_ConnSock_h_
#define _ConnSock_ConnSock_h_

#include <Web/Web.h>

using namespace Upp;

String snd(String r, int a){
 Socket s;
 if(!ClientSocket(s, "127.0.0.1", 5024)) {
    return "x";
 }
 s.Write(r + '\n');
 if(a==1){
        String st = s.ReadUntil('\n');
    return st;
 }
 return "y";
}
#endif
```

if(a==1) a String will be returned by the server, just a comma delimited line, otherwise no ReadUntil.

On the slowest machine Sleep(5) is required between lines or a printout from the server shows some lines are out of sequence or dropped.

Is there an example in upp that shows how to send several lines??

edit: The server is in Java and has:
inRd = new BufferedReader(new InputStreamReader(client.getInputStream()));
I can parse the buffer into lines there.

Neil

Subject: Re: Socket - send multiple lines
Posted by nlneilson on Wed, 10 Nov 2010 20:15:18 GMT
View Forum Message <> Reply to Message

A Google search for "C++ socket send file" returns many examples.
Will C++ compile/run in upp or will there be problems if not wrapped into the upp methods?

Subject: Re: Socket - send multiple lines
Posted by dolik.rce on Wed, 10 Nov 2010 21:10:21 GMT
View Forum Message <> Reply to Message

Hi Neil,

Simple trick I used in similar situation:String Escape(const char* q) {
 String esc;
 for(;*q;q++){
  if(*q=='\\') esc.Cat("\\\\");
  else if(*q=='\n') esc.Cat("\\n");
  else esc.Cat(*q);
 }
 return esc;
}
String Unescape(const char* q){
 String unesc;
 for(;*q;q++) if(*q == '\\'){
  if(*(q+1)=='n'){unesc.Cat('\n');q++;}
  else if(*(q+1)=='\\'){unesc.Cat('\\');q++;}
  else unesc.Cat(*q);
 }else unesc.Cat(*q);
 return unesc;
}

//Example usage:
String manylines="line 1\nline 2\nline 3\n";
String singleline=Escape(manylines); //this can be send using your single line code without major changes

//on the other side just call:
String str=Unescape(s.ReadUntil('\n'));

Note that this is only a workaround and requires you can change source of both client and server, but for me at that time it was the quickest solution

Regarding the question in your other post: You can use any C++ in U++ without problems. Note though, that using C++ sockets directly, you will loose the multiplatform behavior. Also, usually the stuff you need is already available in U++ even if you don't see it at first I am not familiar enough with the socket stuff to really help you, but hopefully someone else here will be able to point you in the right direction...

Best regards,

Subject: Re: Socket - send multiple lines
Posted by nlneilson on Thu, 18 Nov 2010 09:15:12 GMT
View Forum Message <> Reply to Message

Thanks for the quick reply, I left after my last post and just back.

It took a bunch of hours but got it working.

"for(;*q;q++)" is a bit new for me. for(;;) until break I understand. If I debugged some code it would become apparent.

What I did was manually added several lines into char buf including the \n and that worked.

Ln = in.GetLine(); putting that into chbuf for each char dropped the '\n'

```
cc = Ln.GetCount();
if(j==cc || ch=='\n') ch = '\n';
chBuf[kB] = ch;
```

'\n' is just int 10, it wasn't necessary to tinker with escape codes.

Also at the end of a set of lines:
chBuf[kB] = 0;
Otherwise if a previous set was longer there is extra char in the buf.

In TheIde in debug the chbuf just shows the first part, select all or copy didn't work for me.

Data2<<= chBuf; into an EditField the select all->copy worked.
The chbuf held up to ~2100 char.
Then I could paste that into Notepad++ and what was received by the Java app it interacts with to compare.
The C++ to Java is why a socket is necessary. Trying shared memory was a real pain that made my head hurt.

Also what someone may find useful:

```
//        LeftPosZ(13, 248).TopPosZ(64, 54); // for deploy
        LeftPosZ(13, 248).TopPosZ(66, 100); // for debug
```

That way there is room for 5 extra data fields for debugging.

The highest I have tested this is 50 lines per chbuf and sent once every second for one of 60 sets, or when replaying one second represents one minute. The chbuf is made 6o times per second with 50 lines each.
CPU usage <2% running by itself.

edit: I did notice it may take longer than a second but that is not important now. At most in real

time only one chbuf will be made and sent through the socket per second.
For replay of data these are also useful:
pos = in.GetPos();
in.SeekCur(fwd);

No changes to the socket server or client was necessary.

Subject: Re: Socket - send multiple lines
Posted by dolik.rce on Thu, 18 Nov 2010 10:43:17 GMT
View Forum Message <> Reply to Message

Hi Neil,

Good to hear you got it working even with this simple approach

nlneilson wrote on Thu, 18 November 2010 10:15"for(;*q;q++)" is a bit new for me. for(;;) until break I understand. If I debugged some code it would become apparent.
I've learned this from U++ sources. It seems bit weird at first, but it is quite handy once you get used to it and understand the principle. Basically it uses just the fact that *x + n is the same as x[n], only in slightly different syntax

Honza

Subject: Re: Socket - send multiple lines
Posted by nlneilson on Fri, 19 Nov 2010 14:32:18 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Thu, 18 November 2010 11:43
nlneilson wrote on Thu, 18 November 2010 10:15"for(;*q;q++)" is a bit new for me. for(;;) until break I understand. If I debugged some code it would become apparent.

I didn't use that approach. I was not able to read your post until I got back as I have no internet connection at my Ranch/Farm.
That is a major reason the upp installation should have more extensive help data or at least a separate download.
Quote:I'm investigating how to get all the help into theIde at the moment but whether it's achievable or how long it will take to do I don't know yet.
Graeme
  http://www.ultimatepp.org/forum/index.php?t=msg&th=5643& amp; amp; amp;start=0&

As I am not proficient with pointers the "*q" is also something I have not used very much. My programming with C++ was several years ago. Python and Java is what I have used lately which

doesn't use pointers, or at least I don't recall using them in those languages.

I think upp is great, that is why I chose to use it. The way it wraps C++ so it is easier to program reminds me of Python which some consider a wrapper language.
In another upp app I did use pointer to a limited extent:
void split3(String ln, String& S1, String& S2, String& S3);
Maybe that is not even using a pointer.

I started by sending a String that included several lines, 4 at first, that included the \n after each line and that worked.
Then I tried to make a char buffer to match that.

When reading each char of what was to be sent after the last char of a line the \n had to be added. Apparently the in.Getline as far as the length with Ln.GetCount(); counts the \n but the char is not read as '10'(\n) but '0'(null).

With further debugging either of these will work:
if(j==cc) ch='\n; // j+=1 and cc=Ln.GetCount()
if(ch==0) ch='\n;
so my code has:
if(j==cc || ch==0) ch='\n;

In a previous post I mentioned the socket server and client did not have to be changed. When trying another app a glitch was found. The server was changed from reading a String with single line to reading a char buf with many lines. No big deal, just have to change some code.

I am pulling as much of my Java code out of the display app that is based on NASA WWJ and porting it to C++ in upp.
Each of the lines is a point: Latitude, Longitude, Altitude and ID. If the Lat or Lon is out of range of the display (or if the Alt not relevant) the line will be culled or omitted. This is faster in C++ than Java, less heap space problems and is compiled not interpreted at runtime. And C++ is much harder to decompile.

Neil

edit: Honza, looking over your example code:
else if(*q=='\n') esc.Cat("\\n");
As pointed out above the '\n' with Getline() and then reading each char the '\n' is changed to 0. It seems that may be a problem when reading lines from a file, or am I missing something?

Subject: Re: Socket - send multiple lines
Posted by nlneilson on Sat, 20 Nov 2010 22:46:40 GMT
View Forum Message <> Reply to Message

Honza: After a little research I understand where "if(*q=='\n')" can and cannot be used.

Reading a file for each char does read '\n' or whatever for different OS.

C++ "getline", Java "readline" or whatever for different languages is to get the line.

Some languages like C++ drops the '\n' (you don't see it when debugging) and replaces that with the required '\0' to terminate the string. If several lines are to be included in one string the '\0' needs to be replaced with '\n'. The '\0' needs to be retained or added at the end.

In Java the '\n' is read (you can see and manipulate it) but not included in the string unless the code specifically inserts it.

A string is a char buf. The difference on how it is referenced/used is just semantics.

What is sent through my socket is just a bunch of characters. My server code includes the parsing of the data received, if the server just received the data and another function did the parsing no changes to the server code would have been necessary.

I did get my socket server to work with the String/char buf that has a single line or multiple lines. The client just sends what it is told to send, no changes were required there.

Subject: Re: Socket - send multiple lines
Posted by dolik.rce on Sat, 20 Nov 2010 23:24:29 GMT
View Forum Message <> Reply to Message

Hi Neil,

In most cases '\n' is just a character like any other. I think the only cases where it can be dropped is when using "readline" type of functions (such as GetStdIn() or Socket::ReadLine()), at least in U++. Other languages/frameworks might have different traditions.

The '\n' is AFAIK never replaced by '\0'. End of line is a part of string (or char array) and as such it should stay where you put it. Only if the reading procedure drops it, it might appear that '\0' is in its place, but it is actually at the end of any null terminated string (actually there are some cases when a string can end with non-null value, but that is usually weird, confusing and probably not important here).

Honza

Subject: Re: Socket - send multiple lines
Posted by nlneilson on Sun, 21 Nov 2010 03:17:51 GMT
View Forum Message <> Reply to Message

Hi Honza

Most of my experience with C++ was several years ago, the char=0; was not what was expected,

just something to get used to.
If a line in a file is:
ab
Ln = in.GetLine;
Ln[0] = 'a';
Ln[1] = 'b';
Ln[2] = 0;// Java or Python this is '\n'. Note there is no single quotes but if(Ln[2]=='\0'){// is true.

This is in TheIde to make the array before it is sent through the socket.

It is seldom this would be noticed except in a some situations.

Thanks for the help Honza.

Neil

Subject: Re: Socket - send multiple lines
Posted by nlneilson on Mon, 22 Nov 2010 11:30:57 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sun, 21 November 2010 00:24In most cases '\n' is just a character like any other. I think the only cases where it can be dropped is when using "readline" type of functions...
You are correct Honza.
"If the delimiter is found, it is extracted and discarded,..."
http://www.cplusplus.com/reference/string/getline/

Neil
