Subject: How to use StringStream?
Posted by lectus on Tue, 29 Jan 2013 21:51:57 GMT
View Forum Message <> Reply to Message

I csn't find any example of it.

I'm trying to use it together with ZCompress.

Thanks

Subject: Re: How to use StringStream?
Posted by Mindtraveller on Sat, 02 Feb 2013 20:56:02 GMT
View Forum Message <> Reply to Message

What exactly are you trying to do?
If you want to make "streaming" zip/unzip operation, then you may meet some problems here. Anyway, I really tried to use StringStream as simple FIFO buffer for the same thing, and it didn't work as I expected.

So I had to write my own string buffer class which is String-based Stream compatible FIFO-buffer.

This is simplified code how to use it for streaming unzip operation:

```
FileUnZip unzip(filename);
StrIOStream unzipStream; //my simple String-based FIFO class
unzip.ReadFile(unzipStream, callback(this, &MyClass::OnZipData));
while (unzipStream.GetSize())
   ProcessChunk(&unzipStream);
...

bool MyClass::OnZipData(int op1, int op2)
{
   ProcessChunk(&unzipStream);
   return isShutdown; //usually false
}

void MyClass::ProcessChunk(Stream *stream)
{
   String data = stream->Get(static_cast<int>(stream->GetSize())); //get unzipped data
   ... //process the chunk of unzipped data
}
```

The idea is that ProcessChunk() is called from within unzip cycle for streaming. For the last part of unzipped data, the ProcessChunk() is not called automatically, so we call it manually AFTER unzipping.

My tests for the BIG files (about 3 GB in unzipped size) had shown that the maximum size of

unzipped chunk could be ~60 MB. So you have to take it into account.

Finally, if you need it, here is the source code of my Stream-based class:

```
class StrIOStream : public Stream
{
protected:
 virtual  void  _Put(int w)
 {
  data.Cat(w);
 }

 virtual  int   _Term()
 {
  return data.IsEmpty() ? -1 : data[0];
 }

 virtual  int   _Get()
 {
  if (data.IsEmpty())
  return -1;

  int w = data[0];
  data.Remove(0);
  return w;
 }

 virtual  void  _Put(const void *_data, dword size)
 {
  data.Cat((const char *) _data,size);
 }

 virtual  dword _Get(void *_data, dword size)
 {
  dword l = min(size,(dword)data.GetLength());
  memcpy(_data,~data,l);
  data.Remove(0,l);
  return l;
 }

public:
 virtual  void  Seek(int64 pos) {}
 virtual  int64 GetSize() const {return data.GetLength();}
 virtual  void  SetSize(int64 size)
 {
  if (size == data.GetLength())
   return;
  if (size > data.GetLength())
   data.Cat('?', static_cast<int>(size-data.GetLength()));
```

```
  else
   data.Trim(static_cast<int>(size));
 }
 virtual  bool  IsOpen() const {return true;}

protected:
 String         data;

public:
 void        Open(const String& _data) {data = _data;}
 void        Create() {}
 void        Reserve(int n) {data.Reserve(n);}

 String      GetResult() {return data;}
 operator    String()                    { return data; }
};
```