

Hello,

I'm digging the TcpSocket class mostly to learn how it has been done. Something have pinged my attention :

Inet.h

```
enum { BUFFERSIZE = 512 };
enum { NONE, CONNECT, ACCEPT, SSL_CONNECTED };
SOCKET      socket;
int          mode;
char         buffer[BUFFERSIZE];
char         *ptr;
char         *end;
...
int          Get_();
int          Peek_();
int          Peek_(int end_time);
int          Peek(int end_time)      { return ptr < end ? (byte)*ptr : Peek_(end_time); }
```

Socket.c

```
int TcpSocket::Get_()
{
    if(!IsOpen() || IsError() || IsEof() || IsAbort())
        return -1;
    ReadBuffer(GetEndTime());
    return ptr < end ? (byte)*ptr++ : -1;
}
...
void TcpSocket::ReadBuffer(int end_time)
{
    ptr = end = buffer;
    if(Wait(WAIT_READ, end_time))
        end = buffer + Recv(buffer, BUFFERSIZE);
}
```

Maybe the question is stupid, or maybe I missed something. But, Technically, what's the point of having a tiny buffer to receive data when this one is used only in Get() and Peek() and other function to retrieve data (except GetLine()) directly receive from socket ? (ie the function below)

Maybe to perform a software timeout instead of a low level timeout ?

Thanks in advance.

```
int TcpSocket::Get(void *buffer, int count)
{
    LLOG("Get " << count);

    if(!IsOpen() || IsError() || IsEof() || IsAbort())
        return 0;

    int l = (int)(end - ptr);
    done = 0;
    if(l > 0) {
        if(l < count) {
            memcpy(buffer, ptr, l);
            done += l;
            ptr = end;
        }
        else {
            memcpy(buffer, ptr, count);
            ptr += count;
            return count;
        }
    }
    int end_time = GetEndTime();
    while(done < count && !IsError() && !IsEof()) {
        if(!Wait(WAIT_READ, end_time))
            break;
        int part = Recv((char *)buffer + done, count - done);
        if(part > 0)
            done += part;
        if(timeout == 0)
            break;
    }
    return done;
}
```