
Subject: Re: SSL server crash

Posted by [Zbych](#) on Fri, 09 Sep 2011 09:36:09 GMT

[View Forum Message](#) <> [Reply to Message](#)

mirek wrote on Fri, 09 September 2011 10:00

Of course, what you cannot is to take a reference (pointer) to element and then unlock the mutex and use this reference.

But I can do this with Array, can't I?

Can you take a look at this version:

```
#include <Core/Core.h>
```

```
#include <Web/SSL/WebSSL.h>
```

```
using namespace Upp;
```

```
struct client_data: Moveable<client_data> {  
    Socket sock;  
    dword ip;  
    int par1;  
    int par2;  
};
```

```
template <class T>  
class MTAArray: private Array<T>{  
private:  
    Mutex mtx;  
public:  
    T&    Add(const T& x)    {Mutex::Lock __(mtx); Array<T>::Add(x);}  
    void  Remove(int i, int count = 1) {Mutex::Lock __(mtx); Array<T>::Remove(i, count);}  
    const T& operator[](int i) const    {Mutex::Lock __(mtx); return Array<T>::Get(i); }  
    T&    operator[](int i)            {Mutex::Lock __(mtx); return Array<T>::Get(i); }  
    int   GetCount()                  {Mutex::Lock __(mtx); return Array<T>::GetCount(); }  
};
```

```
class TestServer{  
    volatile Atomic stop;  
    MTAArray<client_data> clients;  
  
    int WaitForInput(int _timeout_ms);  
    void Worker();  
public:  
    typedef TestServer CLASSNAME;  
    void Start();  
    void Stop() {AtomicWrite(stop, true);}
```

```
TestServer() {AtomicWrite(stop, false);}
};
```

```
int TestServer::WaitForInput(int timeout_ms)
{
    fd_set set;
    struct timeval tval;
    int max = 0;

    if (clients.GetCount() <= 0) return -1;

    tval.tv_sec = timeout_ms / 1000;
    tval.tv_usec = 1000 * (timeout_ms % 1000);

    FD_ZERO(&set);
    for (int i = 0; i < clients.GetCount(); i++){
        int tmp = clients[i].sock.GetSocket();
        if (tmp > max) max = tmp;
        FD_SET(tmp, &set);
    }

    if (select(max+1, &set, NULL, NULL, &tval) > 0){
        for (int i = 0; i < clients.GetCount(); i++){
            if (FD_ISSET(clients[i].sock.GetSocket(), &set)) return i;
        }
    }

    return -1;
}
```

```
void TestServer::Worker()
{
    while(!Thread::IsShutdownThreads() && !AtomicRead(stop))
    {
        int ci = WaitForInput(1000);
        if (ci >= 0){
            if (!clients[ci].sock.IsOpen() || clients[ci].sock.IsError() || clients[ci].sock.IsEof()){
                Cout() << "Client no " << ci << " (" << FormatIP(clients[ci].ip) << ") closed connection\n";
                clients[ci].sock.Close();
                clients.Remove(ci);
            }else{
                Cout() << "[" << FormatIP(clients[ci].ip) << "]" << clients[ci].sock.Read() << "\n" ;
            }
        }else{
            for (int i = 0; i < clients.GetCount(); i++){
                clients[i].sock.Write(Format("%s: message to client no %d", GetSysTime(), i));
            }
        }
    }
}
```

```

    }
    }
    }
}

void TestServer::Start()
{
    Socket server;
#ifdef 1
    SSLContext context;

    if (!context.Create(SSLv3_server_method())){
        Cout() << "Can not create context\n";
        return;
    }

    if (!context.UseCertificate(LoadFile(ConfigFile("servercert.pem")),
    LoadFile(ConfigFile("serverkey.pem")), false)){
        Cout() << "Certificate and key are different!\n";
        return;
    }

    if(!SSLServerSocket(server, context, 11111, true, 5, true)){
#else
        if(!ServerSocket(server, 11111)){
#endif
            Cout() << "Can not start server\n";
            return;
        }

        Thread().Run(THISBACK(Worker));

        Cout() << "Waiting for connections\n";

        while(!Thread::IsShutdownThreads() && !AtomicRead(stop)){
            client_data new_client;
            if (server.Accept(new_client.sock, &new_client.ip)){
                Cout() << "Client no " << clients.GetCount() << " from " << FormatIP(new_client.ip) << "\n";
                clients.Add(new_client);
            }else{
                Cout() << "+\n";
            }
        }
    }
}

TestServer server;

```

```
void Stop(int sig)
{
    Cout() << "\nstopping, please wait...\n";
    server.Stop();
}
```

```
CONSOLE_APP_MAIN
{
    signal(SIGINT, Stop);
    server.Start();
}
```
