Damian Kaczmarek, Maciej Kamiński

RexIO Terminal Control Library 1.0

Programmer's handbook

for revision number 271

April 2, 2009
www.rexio.org

Preface

RexIO is library for console (rogue-like) user interfaces for applications such as adventure games, full-screen editors, business software etc. providing support for vast variety of terminals and connection types (unified interface for local and remote terms, support for TERMINFO® database and more).

It also provides extensive UI development framework including support for forms, toolbars, subwindows frames etc, full internationalization support including UNICODE^{TM} compliance as well as CJK character properties. Library combines reboustness of modern GUI toolkits and efficiency of console-based IO providing comprehensive solution for virtually any modern software platform.

This paper is an introductory tutorial covering several use cases of this library as well as description of it's basic features, concepts and internat structure. Please refer to enclosed source code listing and reference manual when anything is unclear.

Symbols

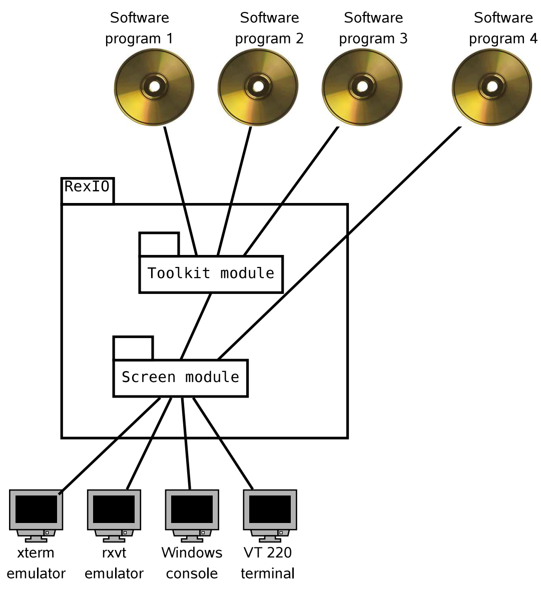
⚠	means, that specific piece of information is important
🧪	indicates a complete program source code, that may be compiled and run
📚	indicates an advanced topic, that is not necessary in basic usage of library

Contents

1 General layout of library

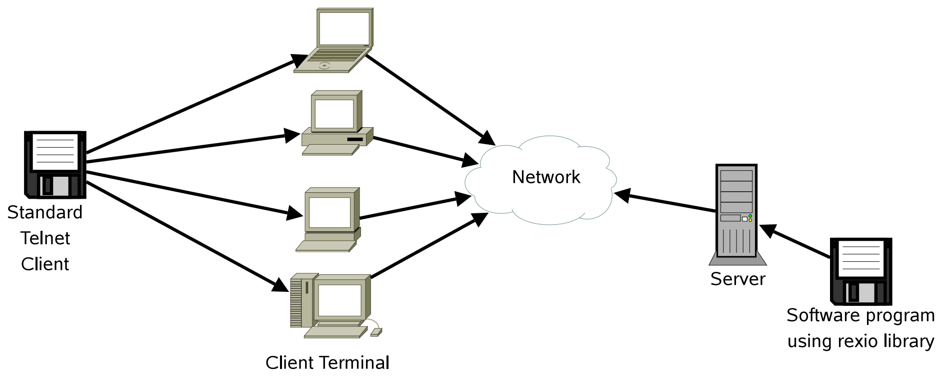
1.1 Possibilities

The library consists of two basic functional blocks: The connectivity module (librexio) and user interface toolkit (librexiotk). The first provides unified interface to screen, while second provides extensive set of extensible utility classes representing some specific interface functionality. Figure below represents typical layout of these items:



As you can see, library aims to provide interface to many different terminal types, and serve as many software applications as possible.

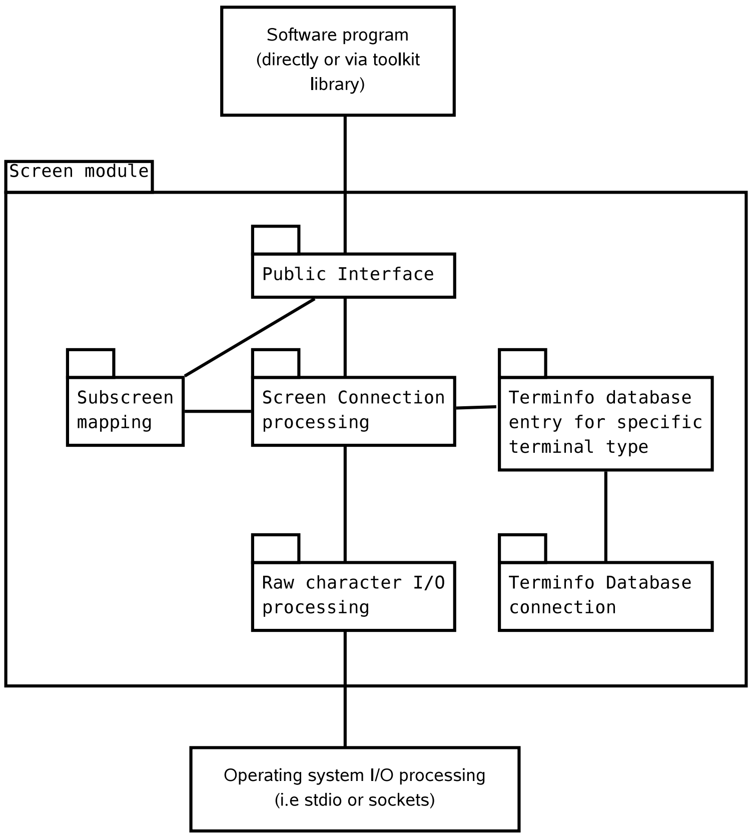
Thanks to being object oriented and thread safe, library may allow single instance of software program communicate with many clients and provide them user interface:



This gives an amazing possibility to create multi user business software, collaborative text editors, and also MMO roleplaying games combining availability of traditional MUD's with easy user interface of rogue like games.

1.2 Internal layout

Layout of functional blocks in screen module is as follows:



Each of them is implemented as set of classes with specific interfaces between them. Some design principles with brief rationale are provided in subsequent sections.

1.2.1 Thread safety

note: „module” symbol in following diagram depicts single instance of specific subsystem (functional block):

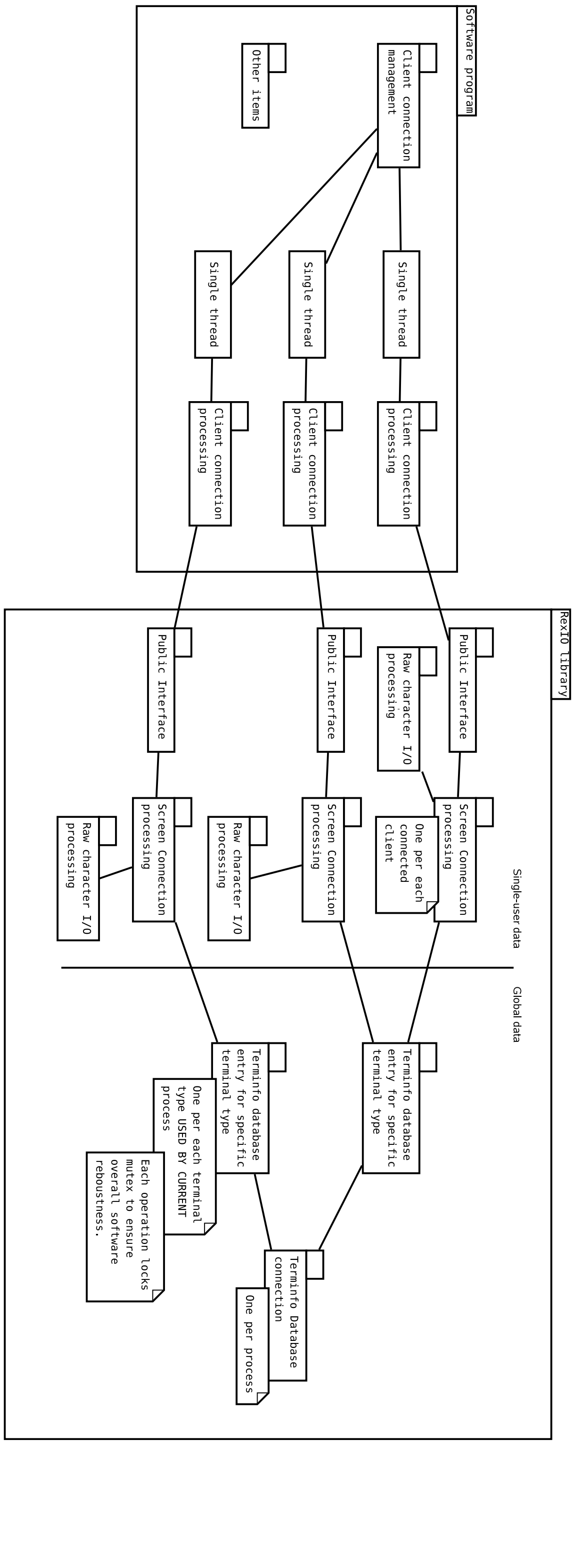

Software program
Client connection management
Other items
Single thread
Single thread
Single thread
Client connection processing
Client connection processing
Client connection processing
RexIO library
Public Interface
Public Interface
Public Interface
Raw character I/O processing
Screen Connection processing
Screen Connection processing
Screen Connection processing
Raw character I/O processing
Raw character I/O processing
One per each connected client
Single-user data
Global data
Terminfo database entry for specific terminal type
Terminfo database entry for specific terminal type
One per each terminal type USED BY CURRENT process
Each operation locks mutex to ensure overall software reboustness.
Terminfo Database connection
One per process

Figure on previous page describes typical layout of multi-threading-connected features, as well as measures taken to obtain stability in threaded programs and ensure their reasonable performance.

As it can be seen, only global data structures are connected to TERMINFO database processing, and other items are separated (one per connection) to achieve reasonable compromise between versatility and efficiency.

General guidelines for writing threaded applications using this library are as follows:

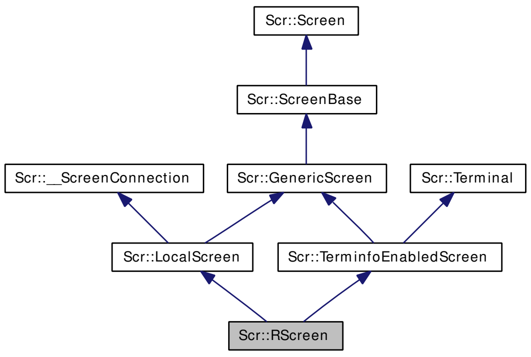
- multiple connections may be created and simultaneously processed in multiple threads (guaranteed by library design).

- multiple operations at the same moment for SINGLE connection are not guaranteed by library design, and therefore special measures must be taken while designing software program that uses this approach.

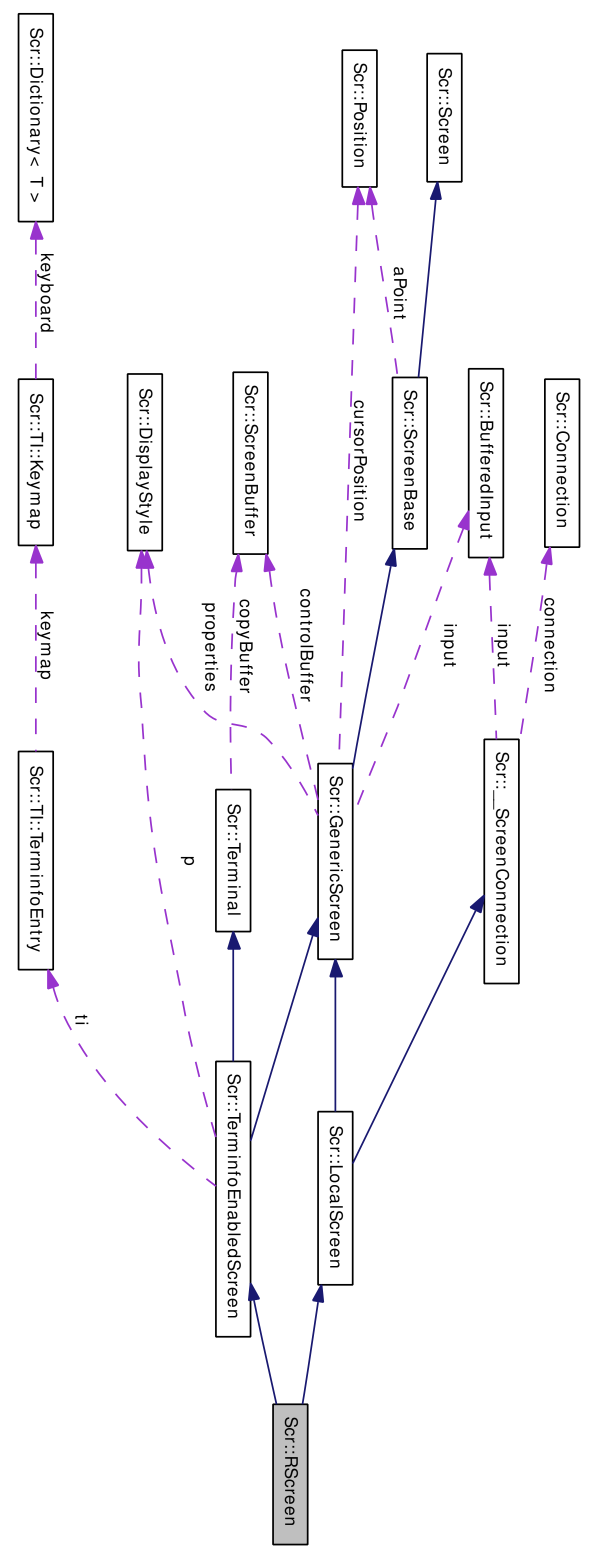
1.2.2 Screen connection processing

Screen class generalizes basic screen operations. Support for different screen and connection types is implemented through inheritance with multiple polymorphism. RScreen (Real Screen Implementation) template generalizes this idea while still allowing to take advantage of OOP (Object Oriented Programming).

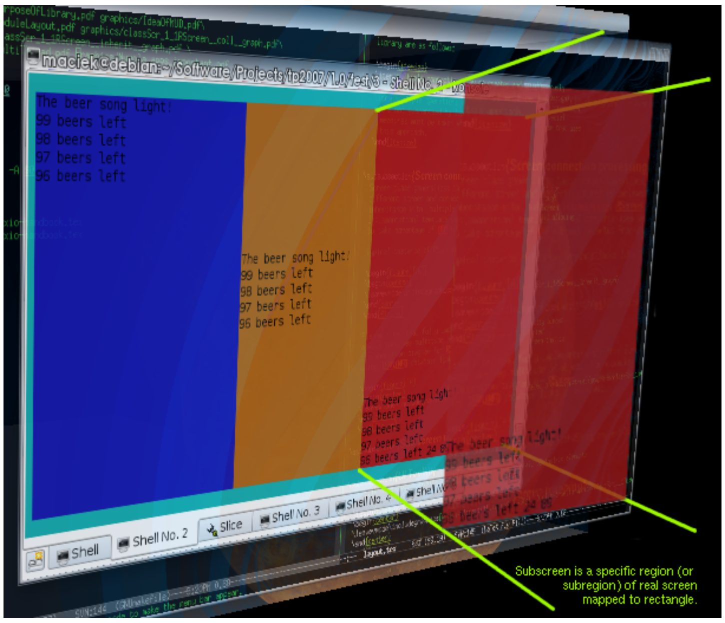
Typical instance of RScreen template looks like this:



Please note, that fully implementing all this functionality almost always involves multitude of object. For example simplified collaboration diagram for RScreen implementing local screen that is using TERMINFO database looks as follows:


Scr::Dictionary< T >
Scr::Position
Scr::Screen
keyboard
aPoint
Scr::TI::Keymap
Scr::DisplayStyle
Scr::ScreenBuffer
cursorPosition
Scr::ScreenBase
Scr::BufferedInput
Scr::Connection
keymap
properties
copyBuffer
controlBuffer
input
input
connection
Scr::TI::TerminfoEntry
p
Scr::Terminal
Scr::GenericScreen
Scr::__ScreenConnection
ti
Scr::TerminfoEnabledScreen
Scr::LocalScreen
Scr::RScreen

1.2.3 Sub screen mapping



```
The beer song light!
99 beers left
98 beers left
97 beers left
96 beers left
```

Shell
Shell No. 2
Slice
Shell No. 3
Shell No. 4
Subscreen is a specific region (or subregion) of real screen mapped to rectangle.

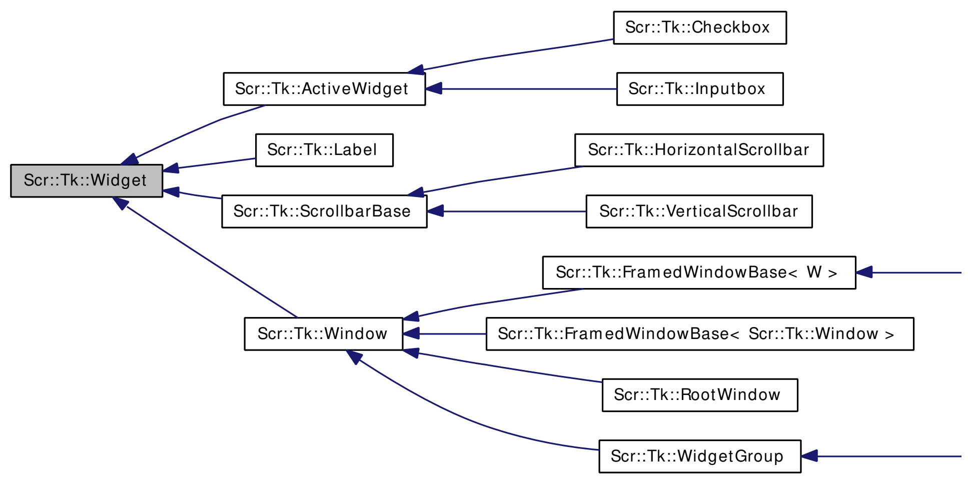
To fully unite interface and provide efficient way of designing hierarchical display structures (as user interfaces) concept of sub-screen is introduced. Sub-screen is section of screen and is itself representative of class screen (inheritance-and-composition pattern).

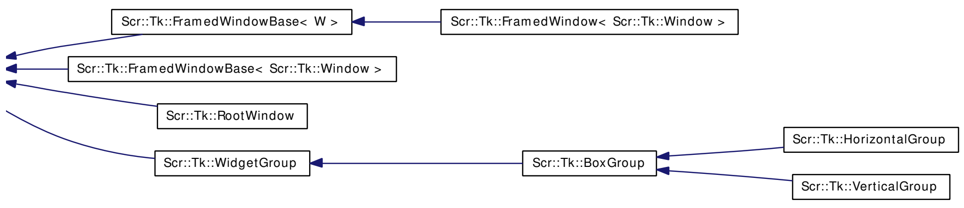
Each basic operation supported by screen is also supported by subscreen, as it inherits its interface. Each subscreen operation is therefore executed on physical screen with appropriate coordinate mapping. Most of these operations affects real screen directly, as subscreen doesn't have its own buffer (therefore active point coordinates of real screen are not preserved after writing on subscreen).

Please refer to Reference Manual for further details.

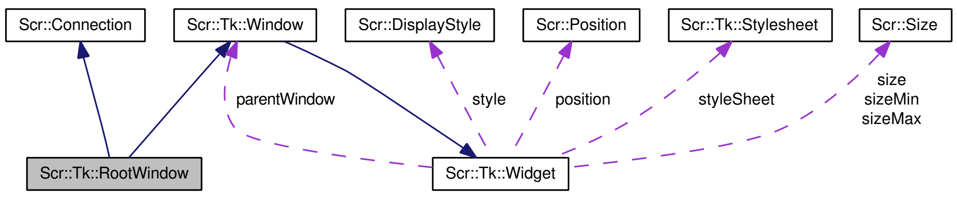
1.2.4 UI Toolkit

Scr::Tk namespace contains Widget class that is base to all UI toolkit elements Following diagrams demonstrate most of Widget descendents:




Scr::Tk::FramedWindowBase< W >
Scr::Tk::FramedWindow< Scr::Tk::Window >
Scr::Tk::FramedWindowBase< Scr::Tk::Window >
Scr::Tk::RootWindow
Scr::Tk::HorizontalGroup
Scr::Tk::WidgetGroup
Scr::Tk::BoxGroup
Scr::Tk::VerticalGroup

Each of them is placed in RootWindow



2 Compiling

To compile this software library you need:

- POSIX compatible operating system
- C++ compiler
- boost libraries (memory and threads)
- cmake build system

When these dependencies are satisfied, you may try to generate UNIX makefile using following command (dot is important - it represents „current path") in root directory.

```
cd **Directory, where RexIO is unpacked**
cmake .
```

When they are finally generated, program is compiled in usual way

```
make
```

When 1.0 release candidate will be ready, install target will be added enabling you to install this library in your system with all other libraries, and use it for all your programs by just typing

```
make install
```

When for some reason you will decide, you do not need this library anymore, you may type

```
make uninstall
```

2.1 Linking your programs against RexIO library

```
g++ yourprogram youroptions -lrexio -lrexiotk
```

3 Screen operations basics

3.1 Typical program layout - message processing pipeline

Typical software program consists of many functional components, that is i.e. some file-reading component, networking subsystem, many other items... and last, but not least, user interface. In object oriented languages, such as C++, many of them are implemented as classes, so no surprise, RexIO is designed to let programmers take advantage of OOP in user interface design.

As a result each UI element is implemented as OBJECT, and each customized element is represented by custom class. Therefore designing user interface is explicitly designing class implementing specific interface. Typical UI design is as follows

```
Specific window type ``MyWindow'' is a RootWindow
  When it is resized,
    it does some action

  When key is pressed,
    it stores it's code if it is a letter
    program quits if it is Esc

  When it has to display its contents
    it displays stored value
```

Design above may be directly converted to C++ code as follows

```cpp
class MyWindow: public RootWindow
{
private:
    int code;
public:
    MyWindow()
        :RootWindow(std::cin,std::cout)
    {
        code=0;
    }


    void OnResize()
    {
        ;//do something
    }

    void  OnKeyDown(Key key)
    {
        if (key == Key::Escape)
            Exit(0);
        else if (key.IsABasicKey())
            code=key;
    }

    void OnRedraw(Scr::Screen &screen)
    {
        screen << Clear << GotoYX(2,2) << code << Refresh;
    }

    ~MyWindow(){;}
};//MyWindow
```

Code above depicts almost complete RexIO application. Full program is as follows:

```
#include<rexio/tk/toolkit.h++>
#include<iostream>
using namespace std;
using namespace Scr;
using namespace Scr::Tk;
using namespace Scr::Control;

class MyWindow: public RootWindow
{
private:
    int code;
public:
    MyWindow()throw()        // empty specification of
                             // throw() means, that function
                             // is not allowed to throw
                             // any exceptions.
        :RootWindow(cin,cout)
    {
        code=0;
    }

    void OnResize()throw()
    {
        ;//do something
        RootWindow::OnResize();
    }

    void  OnKeyDown(Key key)throw()
    {
        if (key == Key::Escape)
            Exit(0);
        else if (key.IsABasicKey())
            code=key;
        RootWindow::OnKeyDown(key);
    }

    void OnRedraw(Scr::Screen &screen)throw()
    {
        try
        {
            screen << Clear << GotoYX(2,2) << code << Refresh;
        }
        catch (...)
        {
            Exit(1);
        }
    }

    ~MyWindow()throw(){;}
};//MyWindow

int main (Uint argc,char ** argv)
{
    RootWindow * app = new MyWindow;
    int result = app->Start(argc, argv);
    delete app;
    return result;
}
/*end of main function of program*/
```

As you can see, line
`#include<rexio/tk/toolkit.h++>`
includes most general library header file (please note, that this file includes virtually „everything" - there are also files declaring specific classes)

lines
```
using namespace std;
using namespace Scr;
using namespace Scr::Tk;
using namespace Scr::Control;
```
aren't necessary to make code working, however they allow to simplify many statements.

Keyword
`throw`
is used in whole library to specify allowed exception sets, and therefore enable controlling exception flow.

Sometimes, when redefining default behavior of windows (especially RootWindow) it is recommended to call default function after (sometimes before) custom processing:
`RootWindow::OnKeyDown(key);`

3.2 Basic character output

In previous section we have discussed basic printing text using following sequence

```
screen << Clear << GotoYX(2,2) << "Hello World" << Refresh;//>>
```

The same effect may be obtained using plain virtual function calls

```
screen.Clear();
screen.GotoYX(2,2);
screen.AddText("Hello World");
screen.Refresh();
```

Please note, that there are multiple (to be precise: 6) variants of AddText: let us consider two of them

```
virtual void AddText(const char * text)
virtual void AddText(const wchar_t * text)
```

One of them accept C-style string with one-byte-per-character, and second accepts wchar_t (for Linux 4 byte, for Windows 2 byte).

The second may be used to print text with diacritics. i.e. to print „Jožin z bažin" you have to type following code:
`screen.AddText(L"Jožin z bažin");`

However as many real software solutions depend on UTF-8 encoding, specific functionality **is provided out of the box**

```
screen.AddText("Jo\xC5\xBEin z ba\xC5\xBEin");
```

does exactly, what you may expect If you want to emphasize "bažin", you may use following code to add colors:

```
screen << "Jo\xC5\xBEin z " << Fg::Bright << Fg::Red << "ba\xC5\
    xBEin";//>>
```

SetFgStyle and SetFgColor functions may be called instead of using this iostream-like syntax. Maybe you won't gain any bigger performance using these functions, but certainly you may improve control of overall layout. Also there are functions like Screen::HorizontalLine simplifying for example box drawing.

Each of these functions provides range checking, and throws specific exception when range is violated. It is recommended to use try-catch statements to detect such problems and provide rock-solid programs that virtually never fail.

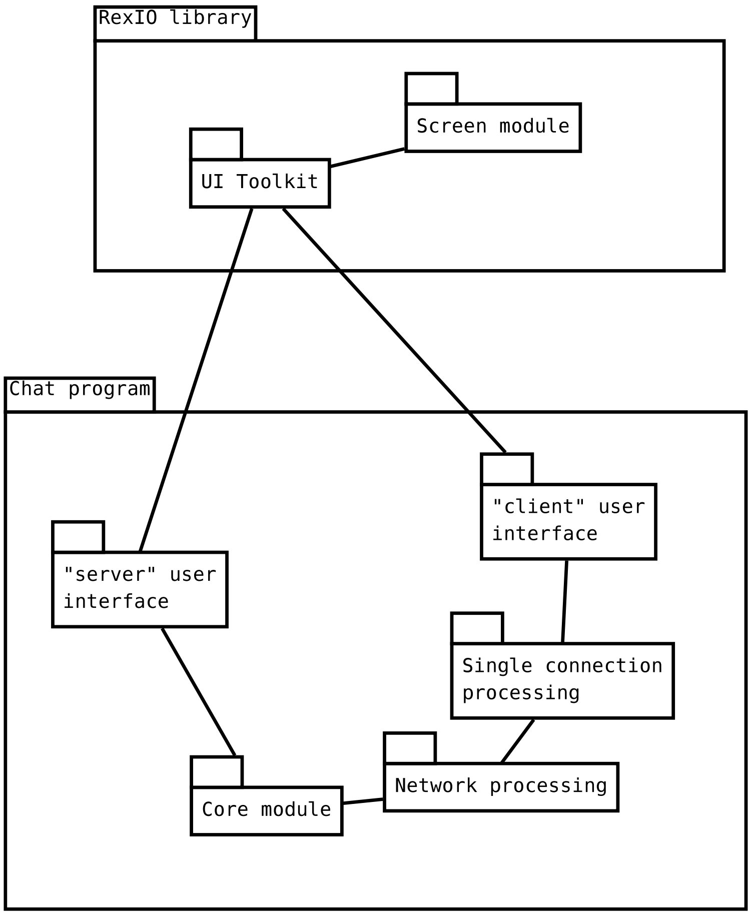
3.3 Component-based hierarchical layout

To improve your understanding of hierarchical layout (basics of Scr::Tk::Widget usage) please consider this piece of code as example.

```
#include<rexio/tk/toolkit.h++>
#include<iostream>
using namespace Scr;
using namespace Scr::Tk;

Scr::Uint labelWidth = 60;

class Demo:public RootWindow
{

public:
    class SampleLabel: public Label
    {public:

        SampleLabel(const std::string& label)throw();
        void OnResize()throw();
    };
    Demo()throw();
    void OnStart()throw();
    void OnResize()throw();
    void OnKeyDown(Key code)throw();
    ~Demo()throw();

    class MultiGroup: public VerticalGroup, public HorizontalGroup
    {
    private:
        bool horizontal;
    public:
        MultiGroup(Uint _height,
                   Uint _width,
                   const DisplayStyle &  _style)throw()
            :BoxGroup(_height, _width,_style),
             VerticalGroup(_height, _width,_style),
             HorizontalGroup(_height, _width,_style),
             horizontal(true)
            {;}
        void ArrangeContents()throw()
            {
                if (horizontal)
                    HorizontalGroup::ArrangeContents();
                else
                    VerticalGroup::ArrangeContents();
            }
        bool GetHorizontal()throw() {return horizontal;}
        void SetHorizontal(bool h)throw() {horizontal=h;
            ArrangeContents();OnResize();}
        void ToggleHorizontal()throw() {horizontal=!horizontal;
            ArrangeContents();OnResize();}
        RTTI_OBJ2(MultiGroup, HorizontalGroup, VerticalGroup);
    };

private:
    MultiGroup *group;
    VerticalGroup *bgroup;
    SampleLabel *blabel[100];
    VerticalGroup *cgroup;
    SampleLabel *clabel[100];
    VerticalGroup *egroup;
    SampleLabel *elabel[100];
    int numLabels;
} app;

Demo::SampleLabel::SampleLabel(const std::string& label)throw()
    :Label(label, DisplayStyle(Fg::Black, Fg::Dark, Bg::
        Transparent))
{;}

void Demo::SampleLabel::OnResize()throw()
{
    ;
}

Demo::Demo()throw()
    :RootWindow(std::cin,std::cout,
                Scr::DisplayStyle(Scr::Fg::White,
                                  Scr::Fg::Bright,
                                  Scr::Bg::Cyan)), numLabels(0)
{
    ;
}

void Demo::OnKeyDown(Key code)throw()
{
    if (code == Key::EoF)
        Exit(1);
    if(code == 'n') {
        group->SwapWidgets(*bgroup, *egroup);
        RedrawRequest();
        return;
    }
    if(code == 'b') {
        group->ShiftBWidget(*cgroup);
        group->RedrawRequest();
        return;
    }
    if(code == 'f') {
        group->ShiftFWidget(*cgroup);
        group->RedrawRequest();
        return;
    }
    if(code == 'v') {
        group->ToggleHorizontal();
        RedrawRequest();
        return;
    }

    if(numLabels == 99)
        return;

    std::stringstream strst;
    strst << 100 - numLabels++<< " beers left";

    blabel[numLabels-1] = new SampleLabel(strst.str());
    bgroup->AddWidget(*blabel[numLabels-1]);
    clabel[numLabels-1] = new SampleLabel(strst.str());
    cgroup->AddWidget(*clabel[numLabels-1]);
    elabel[numLabels-1] = new SampleLabel(strst.str());;
    egroup->AddWidget(*elabel[numLabels-1]);
    RedrawRequest();
}

void Demo::OnStart()throw()
{
    group = new MultiGroup(size.height-2, size.width -2,
                           Scr::DisplayStyle(Scr::Fg::White,
                                             Scr::Fg::Bright,
                                             Scr::Bg::Yellow));

    bgroup = new VerticalGroup(0, 0,
                               Scr::DisplayStyle(Scr::Fg::Red,
                                                 Scr::Fg::Bright,
                                                 Scr::Bg::Blue));
    bgroup->SetAlignPolicy(BoxGroup::Begin);
    cgroup = new VerticalGroup(0, 0,
                               Scr::DisplayStyle(Scr::Fg::Red,
                                                 Scr::Fg::Bright,
                                                 Scr::Bg::
                                                   Transparent));
    cgroup->SetAlignPolicy(BoxGroup::Center);

    egroup = new VerticalGroup(0, 0,
                               Scr::DisplayStyle(Scr::Fg::Yellow,
                                                 Scr::Fg::Dark,
                                                 Scr::Bg::Red));
    egroup->SetAlignPolicy(BoxGroup::End);

    group->AddWidget(*bgroup, 4);
    group->AddWidget(*cgroup, 3);
    group->AddWidget(*egroup, 4);
    AddWidget(*group);
    group->SetPosition(1, 1);

    std::string addstr = "The beer song light!";
    blabel[0] = new SampleLabel(addstr);
    bgroup->AddWidget(*blabel[0]);
    clabel[0] = new SampleLabel(addstr);
    cgroup->AddWidget(*clabel[0]);
    elabel[0] = new SampleLabel(addstr);
    egroup->AddWidget(*elabel[0]);
    numLabels=1;

    RootWindow::OnStart();
}

void Demo::OnResize()throw()
{
    group->SetSize(Size(size.height-2, size.width-2));
    std::cout << size.height-2 << " " <<  size.width-2;

    RootWindow::OnResize();
}

Demo::~Demo()throw()
{
    for (int i = 0 ; numLabels>i ; i++) {
        delete blabel[i];
        delete clabel[i];
        delete elabel[i];
    }
    delete group;
    delete bgroup;
    delete cgroup;
    delete egroup;
}

int main (int argc,char ** argv)
{
    return app.Start();
}
/*end of main function of program*/
```

4 Advanced user interface using Scr::Tk::Widget's

In this chapter we will concern development of basic **useful** software program, that utilizes versatility of RexIO library. It will be an „online chat" program accessed by TELNET.



To run program type for example:

```
./test/5/bin/test  -style=test/5/style.rxs -port=4555
```

`-style`option specifies resource file to be used while processing connections. This resource file specifies not only colours of widgets, but also textual values, so it may be used for internationalization.

4.1 Example program listing

In RexIO distribution this program is located in test/5 directory

4.1.1 include/main.h++

```
#ifndef __MAIN_H__
#define __MAIN_H__
#include <pthread.h>
#include <set>
#include "demo.h++"

class Server;

extern int port;
extern Server s;

class ProgEntry
{
private:
    Scr::Demo* d;
public:
    ProgEntry(Scr::Demo * _d):d(_d){std::cerr << "Adding entry!" << std::endl;}
    Scr::Demo& GetEntry() { return *d; }
    ~ProgEntry()
    {
        d->Exit(4);
        std::cerr << "Deleting entry -" <<std::endl;
    }
    friend bool operator<(const ProgEntry& a, const ProgEntry& b);
};

extern std::set<ProgEntry> allprogs;

void err(const char *s);
extern pthread_mutex_t startSequenceMutex;

#endif
```

4.1.2 include/manager.h++

```
#ifndef __MANAGER_H__
#define __MANAGER_H__

#include <rexio/tk/toolkit.h++>

using namespace Scr;
using namespace Scr::Tk;

class RexLogo:public Widget
{
public:
    void OnRedraw(Screen &scr)throw() {
        scr << GetStyle() << Control::GotoYX(0, 0) <<
            "  ________  ^__^" << Control::GotoYX(1, 0) <<
            " / RexIO? \\ (oo)\\_______" << Control::GotoYX(2, 0) <<
            " ---------- (__)\\       )\\/\\" <<Control::GotoYX(3, 0) <<
            "                ||----w |" <<Control::GotoYX(4, 0) <<
            "                ||     ||" ;
    }
    RexLogo()throw() : Widget(5, 30) { ; }
    RTTI_OBJ(RexLogo, Widget);
};

class WelcomeWindow:public FramedWindow
{
public:
    Label topmsg;
    Label info[16];
    RexLogo logo;

    WelcomeWindow()throw();
    void OnRedrawInside(Screen &scr)throw();
};

class Manager:public RootWindow
{
    WelcomeWindow welcome;
public:
    Manager();
    void OnRedraw(Screen &scr)throw();
    void OnResize()throw();
    void OnStart()throw();
    void OnKeyDown(Key key)throw();
};

#endif // __MANAGER_H__
```

4.1.3 include/demo.h++

```
#ifndef __DEMO_H__
#define __DEMO_H__

#include <rexio/tk/toolkit.h++>
#include "netconn.h++"
#include "manager.h++"

namespace Scr {
    class Demo:public Tk::RootWindow
    {
    public:
        UserInfo userInfo;
    protected:
        class MessageInput : public Tk::Inputbox
        {
        public:
            MessageInput(Uint width)throw();
            void OnKeyDown(Key key)throw();
            void OnFocus(FocusPolicy focustype)throw() {
```

```
        Tk::Inputbox::OnFocus(focustype);
    } // steal focus
};
class LoginWindow : public Scr::Tk::FramedWindow
{
public:
    LoginWindow()throw() :
        FramedWindow(20, 50),
        welcome("Welcome to RexIO chat!"),
        loginInfo("Provide your nickname and press Connect"),
        nameInput(30, L"..Your nickname here.."),
        okButton() {
        objectName="login";
        welcome.objectName="welcome2";
        welcome.SetWidth(46);
        AddWidget(welcome);
        AddWidget(loginInfo);
        loginInfo.SetPosition(2, 0);

        nameInput.SetMaxLength(30);
        nameInput.objectName="nickinput";
        AddWidget(nameInput);
        nameInput.SetPosition(4, 2);
        okButton.objectName="okbutton";
        AddWidget(okButton);
        okButton.SetPosition(4, 34);

        rexioInfo[0].SetText("RexIO is library for console user interfaces.");
        rexioInfo[0].objectName = "rexio1";
        rexioInfo[1].SetText("It provides support for a vast variaty of");
        rexioInfo[1].objectName = "rexio2";
        rexioInfo[2].SetText("terminals and connection types (unified in-");
        rexioInfo[2].objectName = "rexio3";
        rexioInfo[3].SetText("terface for local and remote terms, TERMINFO");
        rexioInfo[3].objectName = "rexio4";
        rexioInfo[4].SetText("and more. See wwww.rexio,org for reference.");
        rexioInfo[4].objectName = "rexio5";

        for(int i = 0;i< 5;i++) {
            AddWidget(rexioInfo[i]);
            rexioInfo[i].SetPosition(7 + i, 0);
            rexioInfo[i].SetSize(1, 47);
            rexioInfo[i].SetStyle(DisplayStyle(Fg::Transparent,
                                               Fg::Bright,
                                               Bg::Transparent));
```

```
                }
                AddWidget(rexlogo);
                rexlogo.SetPosition(12, 8);
            }
            class LoginInput : public Scr::Tk::Inputbox
            {
            public:
                LoginInput(Uint size, const std::wstring &text) throw()
                    : Scr::Tk::Inputbox(size, text) {;};
                bool firstfocus;
                void OnFocus(FocusPolicy focustype)throw() {
                    if(!firstfocus) {
                        firstfocus = true;
                        SetText(L"");
                    }
                    Scr::Tk::Inputbox::OnFocus(focustype);
                }
            };

            class LoginButton : public Scr::Tk::Button
            {
            public:

                LoginButton()throw();

                void OnAction()throw();
                ~LoginButton()throw() {;}

            };

            Scr::Tk::Label welcome;
            Scr::Tk::Label loginInfo;
            LoginInput nameInput;

            LoginButton okButton;

            Scr::Tk::Label rexioInfo[5];
            RexLogo rexlogo;

            RTTI_OBJ(LoginWindow, FramedWindow);
        };
        LoginWindow login;

        class NickList : public Scr::Tk::Window
        {
        public:
            std::list<std::wstring> nicks;

            NickList(Uint _height, Uint _width)throw() :
                Scr::Tk::Window(_height, _width)
                {;}
            void OnRedraw(Screen &scr)throw() {
                Window::OnRedraw(scr);
                try {
                    int cnt = 0;
                    for(std::list<std::wstring>::iterator
                            i = nicks.begin() ; i != nicks.end() ; i++) {
                        scr << Control::GotoYX(cnt++, 0);
                        scr << (*i);
                    }
//          scr << Control::Refresh;
                } catch(...) {
                    ;
                }
            }
            ~NickList()throw(){;};
            RTTI_OBJ(NickList, Window);
        };
        class MsgList : public Scr::Tk::Window
        {
        public:
            std::list<std::wstring> msgs;
            std::list<UserInfo> umsgs;
            Scr::DisplayStyle nickColor;

            MsgList(Uint _height, Uint _width)throw() :
                Scr::Tk::Window(_height, _width)
                {;}
            void OnRedraw(Screen &scr)throw() {
                Window::OnRedraw(scr);
                try {
                    Uint cnt = 0;
                    std::list<UserInfo>::reverse_iterator ui =
                        umsgs.rbegin();
                    for(std::list<std::wstring>::reverse_iterator
                            i = msgs.rbegin() ; i != msgs.rend() ; i++) {
                        if(cnt > GetHeight() - 1)
                            break;

                        scr << Control::GotoYX(GetHeight() - 1 - (cnt++),
                                               0) << nickColor <<
                            (*ui).userName << ": " << GetStyle() << (*i);
                        ui++;
                    }
//          scr << Control::Refresh;
                } catch(...) {
                    ;
                }
            }
            virtual void SetStylesheet(Stylesheet* _styleSheet) throw() {
                Window::SetStylesheet(_styleSheet);
                __FetchProperty(nickColor, "nickColor");
```

```
            }
            ~MsgList()throw(){;};
            RTTI_OBJ(MsgList, Window);
        };

        Scr::Tk::VerticalGroup maing;

        Scr::Tk::Label infoBar;
        Scr::Tk::HorizontalGroup centerg;
        MessageInput msgInput;

        MsgList msgList;
        NickList nickList;

        std::vector<Label> nicklist;
        std::vector<Label> msglist;
    public:
        Demo(std::istream & in, std::ostream & out)throw();
        void OnResize()throw();
        void OnStart()throw();
        void MessageEvent(const UserInfo& info,
                          const std::wstring& msg)throw();
        void JoinEvent(const UserInfo& info)throw();
        void LeaveEvent(const UserInfo& info)throw();

        ~Demo()throw();
    };
}
#endif
```

4.1.4 include/netconn.h++

```
#ifndef __NETCONN_H__
#define __NETCONN_H__
#include <rexio/screen.h++>
#include <list>

typedef void (*ConnectionFunc)(std::istream & in, std::ostream & out) ;

class UserInfo
{
public:
    UserInfo(const std::wstring& user)throw() : userName(user) {
        ;
    }
    UserInfo()throw() {;};
    std::wstring userName;
    Scr::DisplayStyle userColor;
};

class Server
{

```

```
private:
    bool active;
public:
    Server();
    std::list<std::wstring> nicks;

    void Start(int portnum, ConnectionFunc _f);
    void Stop();

    void MessageEvent(const UserInfo& info,
                      const std::wstring& msg)throw();
    void JoinEvent(const UserInfo& info)throw();
    void LeaveEvent(const UserInfo& info)throw();
};
#endif
```

4.1.5 src/main.c++

```
#include <iostream>
#include <exception>
#include <signal.h>
#include "netconn.h++"
#include "demo.h++"
#include "main.h++"
#include "manager.h++"
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <set>
#include <cstdlib>
#include <cstring>

pthread_mutex_t startSequenceMutex;
pthread_mutex_t allprogsStackMutex;

using namespace std;

// for set<ProgEntry> underlaying tree.
bool operator<(const ProgEntry& a, const ProgEntry& b) {
    // comparing addresses of a and b would be wrong, as they may
    // differ while referencing the same Demo class object.
    return a.d < b.d;
}

set<ProgEntry> allprogs;
Server s;

static std::pair<int, char **> args;

void starter(std::istream & in, std::ostream & out)
{
    Demo prog(in,out);
    pthread_mutex_lock(&allprogsStackMutex);// prevent accidental stack
    allprogs.insert(ProgEntry(&prog));// data structure destruction
    pthread_mutex_unlock(&allprogsStackMutex);
    cerr << "Trying to initialize connection" << endl;
    try
    {
        int i = prog.Start(args.first, args.second); // start
        cerr << "Connection finished with code " << i <<endl;// result
        // on success
    }
    catch (exception)// exception caught. try to recover by ignoring it.
    {
        cerr << "Connection finished with exception, but app is fine."
             <<endl;
    }
    pthread_mutex_lock(&allprogsStackMutex);
    cerr << "Requesting erase of 1 app out of "<<allprogs.size()<< endl;;

    // if ProgEntry exists (not deleted by localInterface func)
    if (allprogs.find(ProgEntry(&prog)) != allprogs.end())
        allprogs.erase(ProgEntry(&prog));//erase it.
    cerr << endl;
    pthread_mutex_unlock(&allprogsStackMutex);
    return;
}

Manager manager;

void * localInterface(void * arg)
{
    std::pair<int, char **>& args = *
        reinterpret_cast<std::pair<int, char **> *>(arg);

    manager.Start(args.first, args.second);
// as login as manager is running, everything is
    // fine. Start shutdown procedure when it stops.


    s.Stop();
    cerr << "Server stopped correctly. ";
    pthread_mutex_lock(&allprogsStackMutex);
    cerr << "Requesting " << allprogs.size()<<" client apps to end ."<<endl;
    allprogs.clear();//stop all instances of program  (ProgEntry
             //destructor stops associated app)
    cerr << endl;
    pthread_mutex_unlock(&allprogsStackMutex);
    return 0;
```

```
}

//used in netconn and other
void err(const char * s)
{
    manager.Exit(0);
    sleep(3);// make sure, it'll be displayed afted last message
    cerr << "\nFatal error occured:" <<s << endl;
    exit(1);
}

typedef void (*pfv)();
int port = 5000;
int main (int argc,char ** argv)
{
    args.first = argc;
    args.second = argv;

    if(argc > 1)
    {
        stringstream ss;
        ss<<argv[1];

        for(int i = 0;i<argc;i++) {
            if(strncmp(argv[i], "-port=", 6) == 0) {
                std::string str(argv[i] + 6);
                std::stringstream ss(str);
                ss >> port;
            }
        }
    }

    pthread_t ctl_local;
    pthread_mutex_init(&startSequenceMutex,NULL);
    pthread_mutex_lock(&startSequenceMutex); // unlocked after
                          // clearing screen by
                          // Manager object
    pthread_create(&ctl_local, NULL, localInterface, new std::pair
        <int, char**>(argc, argv));
    pthread_mutex_lock(&startSequenceMutex);
    pthread_mutex_unlock(&startSequenceMutex);
    pthread_mutex_destroy(&startSequenceMutex);


    pthread_mutex_init(&allprogsStackMutex,NULL);

    cerr << "opening port " << port << endl;

    signal(SIGPIPE, SIG_IGN);   //disable signal (app has other
        ways
    //of detecting connection errors)
    s.Start(port,starter);
    pthread_join(ctl_local,NULL);
    cerr << "Game over" << endl;
    pthread_mutex_destroy(&allprogsStackMutex);
    return 0;
}
/*end of main function of program*/
```

4.1.6 src/manager.c++

```
#include <rexio/screen.h++>
#include "manager.h++"
#include "main.h++"
#include <iostream>
#include <iomanip>

using namespace Scr;
using namespace Scr::Tk;

WelcomeWindow::WelcomeWindow()throw()
    :FramedWindow(20, 50), topmsg("Welcome to RexIO demo application!")
//,
//                                  DisplayStyle(Fg::Red, Fg::Dark, Bg::Yellow),
//                                  FrameStyle(DisplayStyle(Fg::Red, Fg::Dark, Bg::Green)))
{
    topmsg.objectName="welcome";
    AddWidget(topmsg);
    topmsg.SetPosition(1, 1);

    info[0].SetText("This demo will show a sample network chat");
    info[1].SetText("application with console user interface");
    info[2].SetText("streamed over ordinary telnet application.");
    info[3].SetText("");
    info[4].SetText("");
    info[6].SetText("Use telnet to connect to the above port.");
    info[7].SetText("");
    info[8].SetText("");
    info[9].SetText("");
    info[10].SetText("");
    info[11].SetText("");
    info[12].SetText("");
    info[13].SetText("");
    info[14].SetText("");
    info[15].SetText("Enjoy!");

    for(int i = 0;i< 16;i++) {
        info[i].SetStyle(DisplayStyle(Fg::Transparent, Fg::Dark,
                                      Bg::Transparent));
        AddWidget(info[i]);
        info[i].SetPosition(i + 3, 1);
        info[i].SetSize(1, 47);
        std::stringstream ss;
        ss << "info" << i;
        info[i].objectName = ss.str();
    }

    AddWidget(logo);
    logo.SetPosition(11, 9);
}

void WelcomeWindow::OnRedrawInside(Scr::Screen &scr)throw()
{
    FramedWindow::OnRedrawInside(scr);
//  scr << Control::Refresh;
}

Manager::Manager(): RootWindow(std::cin, std::cout),
                    welcome()
{

    AddWidget(welcome);
}

void Manager::OnResize()throw()
{
    try
    {
    welcome.SetPosition((GetHeight() - welcome.GetHeight())/2,
                        (GetWidth() - welcome.GetWidth())/2);
    }catch(...){;}// exception may be thrown if OnResize called
        before OnStart()
}

void Manager::OnRedraw(Scr::Screen &scr)throw()
{
    RootWindow::OnRedraw(scr);
}

void Manager::OnStart()throw()
{
    std::stringstream ss;
    ss << ::port;
    welcome.info[5].SetText("Port: " + ss.str());

    pthread_mutex_unlock(&startSequenceMutex);
}

void Manager::OnKeyDown(Scr::Key key)throw()
{
    if (key=='q') Exit(0);
}
```

4.1.7 src/demo.c++

```
#include "demo.h++"
#include "netconn.h++"
#include "main.h++"

```

```
using namespace Scr;
using namespace Scr::Tk;

Demo::Demo(std::istream & in,
           std::ostream & out)throw() :
    RootWindow(in, out),
    maing(GetHeight(), GetWidth()),
    infoBar("RexIO chat application."),
    centerg(GetHeight() - 2, GetWidth()),
    msgInput(GetWidth()),
    msgList(0, 0),
    nickList(0, 0)
{
    infoBar.objectName = "infobar";
    msgList.objectName = "msglist";
    nickList.objectName = "nicklist";
    msgInput.objectName = "msginput";
}

void Demo::MessageInput::OnKeyDown(Key key)throw()
{
    if(key.IsASpecialKey())
        if(key.GetSpecialKey() == Key::Enter) {
            s.MessageEvent(static_cast<Demo&>(GetParent().
                GetRootWindow()).userInfo, GetText());
            SetText(L"");
            return;
        }
    SetActive(true);
    Inputbox::OnKeyDown(key);
}

Demo::MessageInput::MessageInput(Uint width)throw() : Inputbox(
    width, L"")
{
    SetMaxLength(100);
}

Demo::LoginWindow::LoginButton::LoginButton()throw() :
    Scr::Tk::Button(1, 12, "Connect")
{
    ;
}

void Demo::LoginWindow::LoginButton::OnAction()throw()
{
    Demo &demo = static_cast<Demo&>
        (GetParent().GetRootWindow());
    demo.maing.SetHidden(false);
    demo.login.SetHidden(true);
    demo.OnFocus(TabFocus);
    demo.userInfo = UserInfo(demo.login.nameInput.GetText());
    s.JoinEvent(demo.userInfo);
}
```

```

void Demo::OnResize()throw()
{
    RootWindow::OnResize();
    maing.SetSize(Size(GetHeight(), GetWidth()));
    msgInput.SetSize(1, GetWidth());
    if(elements[&login] != elements.end())
        login.SetPosition((GetHeight() - login.GetHeight())/2,
                          (GetWidth() - login.GetWidth())/2);
}

void Demo::OnStart()throw()
{
    AddWidget(login);

    AddWidget(maing);
    maing.SetHidden(true);
    maing.AddWidget(infoBar);
    maing.AddWidget(centerg);
    maing.AddWidget(msgInput);

    centerg.AddWidget(msgList, 4);
    centerg.AddWidget(nickList);
}

void Demo::MessageEvent(const UserInfo &info,
                        const std::wstring& msg)throw()
{
    msgList.msgs.push_back(msg);
    msgList.umsgs.push_back(info);
    RedrawRequest();
}

void Demo::JoinEvent(const UserInfo& info)throw()
{
    nickList.nicks = s.nicks;
    RedrawRequest();
}

void Demo::LeaveEvent(const UserInfo& info)throw()
{
    nickList.nicks.remove(info.userName);
    s.nicks.push_back(info.userName);
    RedrawRequest();
}

Demo::~Demo()throw()
{
    s.LeaveEvent(userInfo);
}
```

4.1.8 src/netconn.c++

```
#include <iostream>
```

```
#include <pthread.h>
#include <fcntl.h>
#include "main.h++"
#include "netconn.h++"
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <ext/stdio_filebuf.h>
#include <unistd.h> /* sleep*/
#include <stack>
#include <cstring>

#include<ext/stdio_filebuf.h>

using namespace std;

/*Class for internal use representing reprezenting initialization
 and termination of connection*/
class __Connection
{
private:
    int fd;
    pthread_t th;

    FILE * oF;
    __gnu_cxx::stdio_filebuf<char> * obuf;
    ostream * ostr;

    FILE * iF;
    __gnu_cxx::stdio_filebuf<char> * ibuf;
    istream * istr;
public:
    __Connection(int _fd);
    ~__Connection();    friend void * ServeConnnection(void * conn
        );
};
/* connection */

/* callback function serving connection*/
ConnectionFunc f;

/*for joining "dead" threads*/
stack<pthread_t> CleanerStack;
pthread_mutex_t CleanerStackMutex;

void * ServeConnnection(void * _conn)
{
    __Connection * conn = (__Connection *) _conn;
    cerr << "in thread for conn fd: "<< conn->fd << endl;

    f(*(conn->istr),*(conn->ostr));
    delete conn;
    return 0;
}
```

```

__Connection::__Connection(int _fd)
{
    fd=_fd;

    cerr << "Accepted connection; fd = " << fd << endl;
    iF = fdopen(fd,"r");
    oF = fdopen(fd,"w");
    ibuf = new __gnu_cxx::stdio_filebuf<char>(iF,std::ios_base::in
        ,1);
    obuf = new __gnu_cxx::stdio_filebuf<char>(oF,std::ios_base::
        out,1);
    istr = new std::istream(ibuf);
    ostr = new std::ostream(obuf);

    pthread_create(&th, NULL, ServeConnnection, this);
}

__Connection::~__Connection()
{
    delete istr;
    delete ibuf;
    fclose(iF);

    delete ostr;
    delete obuf;
    fclose(oF);
    close(fd);
    pthread_mutex_lock(&CleanerStackMutex);
    CleanerStack.push(th);
    pthread_mutex_unlock(&CleanerStackMutex);
}

void * CleanerFunc(void * activity_mark)
{
    while (* static_cast<bool*>(activity_mark))
    {
        sleep(2);
        pthread_mutex_lock(&CleanerStackMutex);
        while (!CleanerStack.empty())
        {
            pthread_t t = CleanerStack.top();
            pthread_join(t,NULL);
            CleanerStack.pop();
            cerr << "Joined thread" << endl;
        }
        pthread_mutex_unlock(&CleanerStackMutex);
    }
    return 0;
}


Server::Server() {;}
void Server::Start(int portnum,ConnectionFunc _f){

    f=_f;
    active=true;
    struct sockaddr_in srv;
    socklen_t socklen;
    int iSockFD;
    if ((iSockFD=socket(PF_INET,SOCK_STREAM,0))<0)
        err("socket");

    int opt = 1, len = 4;
    setsockopt(iSockFD, SOL_SOCKET, SO_REUSEADDR, &opt, len);

    memset(&srv,0,sizeof(srv));
    srv.sin_family = AF_INET;
    srv.sin_addr.s_addr = htonl(INADDR_ANY);
    srv.sin_port = htons(portnum);

    socklen=sizeof(srv);

    if (bind(iSockFD, (struct sockaddr *) & srv, socklen) < 0)
        err("bind");

    struct sockaddr_in cli;
    int fd;
    listen(iSockFD,5);

    if(pthread_mutex_init(&CleanerStackMutex,NULL))
        err("mutex_initialize");
    pthread_t cleaner_thread;

    if (fcntl(iSockFD, F_SETFL, O_NDELAY) < 0)
        err("Can't make nonblocking socked");

    if (pthread_create (&cleaner_thread,NULL,CleanerFunc,&active))
        err("pthread_create (&cleaner_thread,NULL,CleanerFunc,NULL
            )");

    while (active)
    {
        fd = accept(iSockFD, (struct sockaddr *) & cli, & socklen)
            ;
        if (fd>0)
            new __Connection(fd);
        else
            usleep(1000);
    }
    if (pthread_join(cleaner_thread,NULL))
        err("pthread_join(&cleaner_thread,NULL)");
    pthread_mutex_destroy(&CleanerStackMutex);
    close(iSockFD);
}

void Server::MessageEvent(const UserInfo& info,
                          const std::wstring& msg)throw()
```

```
{
    for(set<ProgEntry>::iterator i = allprogs.begin(); i !=
        allprogs.end();
        i++) {
        Scr::Demo &target = const_cast<ProgEntry &>((*i)).GetEntry
            ();
        target.MessageEvent(info, msg);
    }
}

void Server::JoinEvent(const UserInfo& info)throw()
{
    s.nicks.push_back(info.userName);
    for(set<ProgEntry>::iterator i = allprogs.begin(); i !=
        allprogs.end();
        i++) {
        Scr::Demo &target = const_cast<ProgEntry &>((*i)).GetEntry
            ();
        target.JoinEvent(info);
    }
}

void Server::LeaveEvent(const UserInfo& info)throw()
{
    for(set<ProgEntry>::iterator i = allprogs.begin(); i !=
        allprogs.end();
        i++) {
        Scr::Demo &target = const_cast<ProgEntry &>((*i)).GetEntry
            ();
        if( &target.userInfo != &info)
            target.LeaveEvent(info);
    }
    s.nicks.remove(info.userName);
}

void Server::Stop()
{
    active=false;
}
```

4.1.9 src/netconn.c++

```
#include <iostream>
#include <pthread.h>
#include <fcntl.h>
#include "main.h++"
#include "netconn.h++"
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <ext/stdio_filebuf.h>
#include <unistd.h> /* sleep*/
#include <stack>
#include <cstring>
```

```

#include<ext/stdio_filebuf.h>

using namespace std;

/*Class for internal use representing reprezenting initialization
 and termination of connection*/
class __Connection
{
private:
    int fd;
    pthread_t th;

    FILE * oF;
    __gnu_cxx::stdio_filebuf<char> * obuf;
    ostream * ostr;

    FILE * iF;
    __gnu_cxx::stdio_filebuf<char> * ibuf;
    istream * istr;
public:
    __Connection(int _fd);
    ~__Connection();    friend void * ServeConnnection(void * conn
        );
};
/* connection */

/* callback function serving connection*/
ConnectionFunc f;

/*for joining "dead" threads*/
stack<pthread_t> CleanerStack;
pthread_mutex_t CleanerStackMutex;

void * ServeConnnection(void * _conn)
{
    __Connection * conn = (__Connection *) _conn;
    cerr << "in thread for conn fd: "<< conn->fd << endl;

    f(*(conn->istr),*(conn->ostr));
    delete conn;
    return 0;
}

__Connection::__Connection(int _fd)
{
    fd=_fd;

    cerr << "Accepted connection; fd = " << fd << endl;
    iF = fdopen(fd,"r");
    oF = fdopen(fd,"w");
    ibuf = new __gnu_cxx::stdio_filebuf<char>(iF,std::ios_base::in
        ,1);
    obuf = new __gnu_cxx::stdio_filebuf<char>(oF,std::ios_base::
        out,1);
    istr = new std::istream(ibuf);
    ostr = new std::ostream(obuf);

    pthread_create(&th, NULL, ServeConnnection, this);
}

__Connection::~__Connection()
{
    delete istr;
    delete ibuf;
    fclose(iF);

    delete ostr;
    delete obuf;
    fclose(oF);
    close(fd);
    pthread_mutex_lock(&CleanerStackMutex);
    CleanerStack.push(th);
    pthread_mutex_unlock(&CleanerStackMutex);
}

void * CleanerFunc(void * activity_mark)
{
    while (* static_cast<bool*>(activity_mark))
    {
        sleep(2);
        pthread_mutex_lock(&CleanerStackMutex);
        while (!CleanerStack.empty())
        {
            pthread_t t = CleanerStack.top();
            pthread_join(t,NULL);
            CleanerStack.pop();
            cerr << "Joined thread" << endl;
        }
        pthread_mutex_unlock(&CleanerStackMutex);
    }
    return 0;
}


Server::Server() {;}
void Server::Start(int portnum,ConnectionFunc _f){

    f=_f;
    active=true;
    struct sockaddr_in srv;
    socklen_t socklen;
    int iSockFD;
    if ((iSockFD=socket(PF_INET,SOCK_STREAM,0))<0)
        err("socket");

    int opt = 1, len = 4;
    setsockopt(iSockFD, SOL_SOCKET, SO_REUSEADDR, &opt, len);

    memset(&srv,0,sizeof(srv));
    srv.sin_family = AF_INET;
    srv.sin_addr.s_addr = htonl(INADDR_ANY);
    srv.sin_port = htons(portnum);

    socklen=sizeof(srv);

    if (bind(iSockFD, (struct sockaddr *) & srv, socklen) < 0)
        err("bind");

    struct sockaddr_in cli;
    int fd;
    listen(iSockFD,5);

    if(pthread_mutex_init(&CleanerStackMutex,NULL))
        err("mutex_initialize");
    pthread_t cleaner_thread;

    if (fcntl(iSockFD, F_SETFL, O_NDELAY) < 0)
        err("Can't make nonblocking socked");

    if (pthread_create (&cleaner_thread,NULL,CleanerFunc,&active))
        err("pthread_create (&cleaner_thread,NULL,CleanerFunc,NULL)");

    while (active)
    {
        fd = accept(iSockFD, (struct sockaddr *) & cli, & socklen);
        if (fd>0)
            new __Connection(fd);
        else
            usleep(1000);
    }
    if (pthread_join(cleaner_thread,NULL))
        err("pthread_join(&cleaner_thread,NULL)");
    pthread_mutex_destroy(&CleanerStackMutex);
    close(iSockFD);
}

void Server::MessageEvent(const UserInfo& info,
                          const std::wstring& msg)throw()
{
    for(set<ProgEntry>::iterator i = allprogs.begin(); i != allprogs.end();
        i++) {
        Scr::Demo &target = const_cast<ProgEntry &>((*i)).GetEntry();
        target.MessageEvent(info, msg);
    }
}

```

```
void Server::JoinEvent(const UserInfo& info)throw()
{
    s.nicks.push_back(info.userName);
    for(set<ProgEntry>::iterator i = allprogs.begin(); i !=
        allprogs.end();
        i++) {
        Scr::Demo &target = const_cast<ProgEntry &>((*i)).GetEntry
            ();
        target.JoinEvent(info);
    }
}

void Server::LeaveEvent(const UserInfo& info)throw()
{
    for(set<ProgEntry>::iterator i = allprogs.begin(); i !=
        allprogs.end();
        i++) {
        Scr::Demo &target = const_cast<ProgEntry &>((*i)).GetEntry
            ();
        if( &target.userInfo != &info)
            target.LeaveEvent(info);
    }
    s.nicks.remove(info.userName);
}

void Server::Stop()
{
    active=false;
}
```

4.1.10 style.rxs

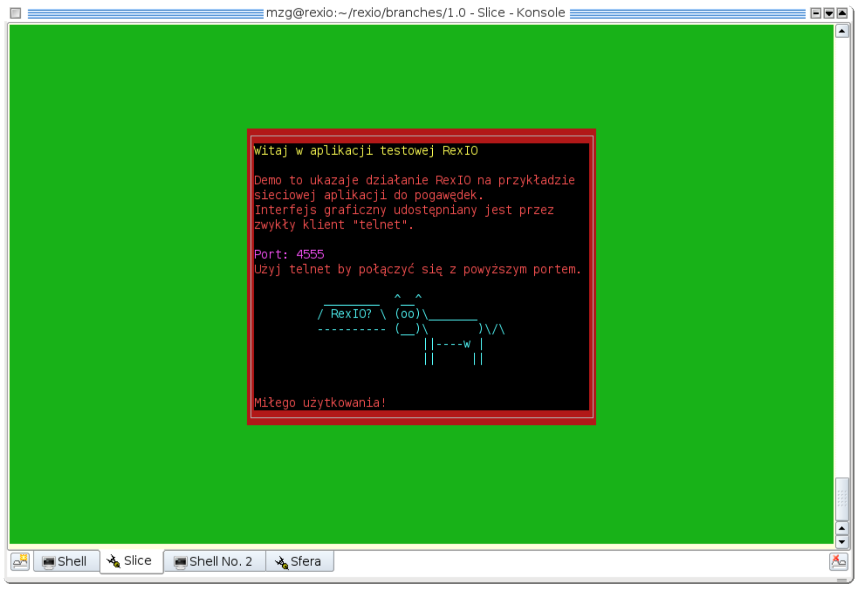
```
RootWindow {
    style: Black Dark Green;
}

FramedWindow {
    style: Red Bright Black;
    frameColor: White Dark Red;
}

LoginWindow {
    style: Red Bright Black;
    frameColor: White Dark Red;
}

Label#welcome {
    style: Yellow Bright Transparent;
    content: "Witaj w aplikacji testowej RexIO";
}
```

```
Label#welcome2 {
    style: Yellow Bright Transparent;
    content: "Witaj w aplikacji do pogawędek RexIO!";
}

Inputbox#nickinput {
    style: White Dark Blue;
    cursorStyle: Yellow Bright Yellow;
    activeStyle: White Bright Blue;
}

Button#okbutton {
    style: White Dark Blue;
    activeStyle: White Bright Blue;
}

Label#info0 { content: "Demo to ukazaje działanie RexIO na przykładzie";}
Label#info1 { content: "sieciowej aplikacji do pogawędek.";}
Label#info2 { content: "Interfejs graficzny udostępniany jest przez";}
Label#info3 { content: "zwykły klient "telnet".";}
Label#info5 { style: Magenta Bright Transparent; }
Label#info6 { content: "Użyj telnet by połączyć się z powyższym portem.";}

Label#info15 { content: "Miłego użytkowania!"; }

Label#rexio1 { content: "RexIO jest biblioteką kontroli interfejsu";}
Label#rexio2 { content: "tekstowego z wsparciem dla szerokiej gamy";}
Label#rexio3 { content: "terminali oraz sposobów łączenia. Terminale";}
Label#rexio4 { content: "zdalne jak i lokalne obsługiwane są przez";}
Label#rexio5 { content: "klasy o takim samym interfejsie.";}

RexLogo { style: Cyan Bright Transparent; }

Label#infobar {
    style: White Bright Yellow;
}

Inputbox#msginput {
    style: White Dark Blue;
    cursorStyle: Yellow Bright Yellow;
    activeStyle: White Bright Blue;
}
```

```
NickList {
    style: Yellow Bright Red;
    nickColor: Magenta Bright Transparent;
}
```

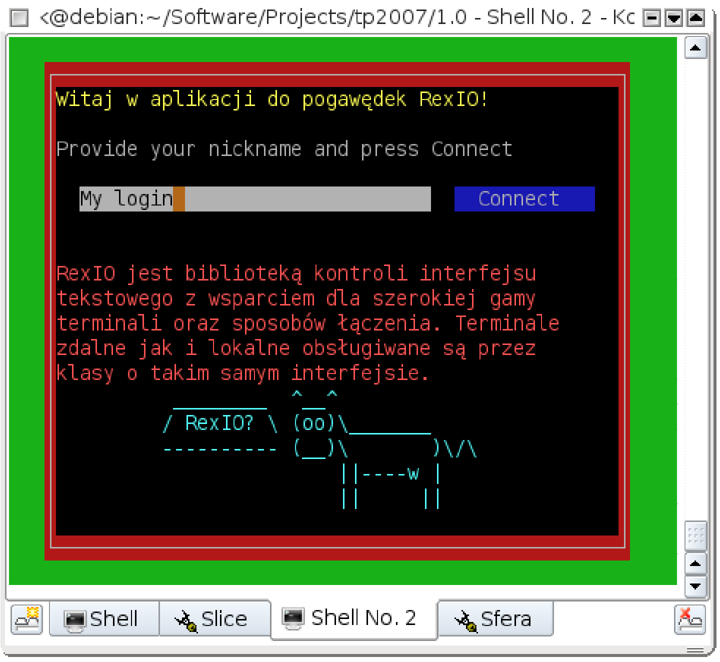
4.2 Screenshots

Server welcome screen



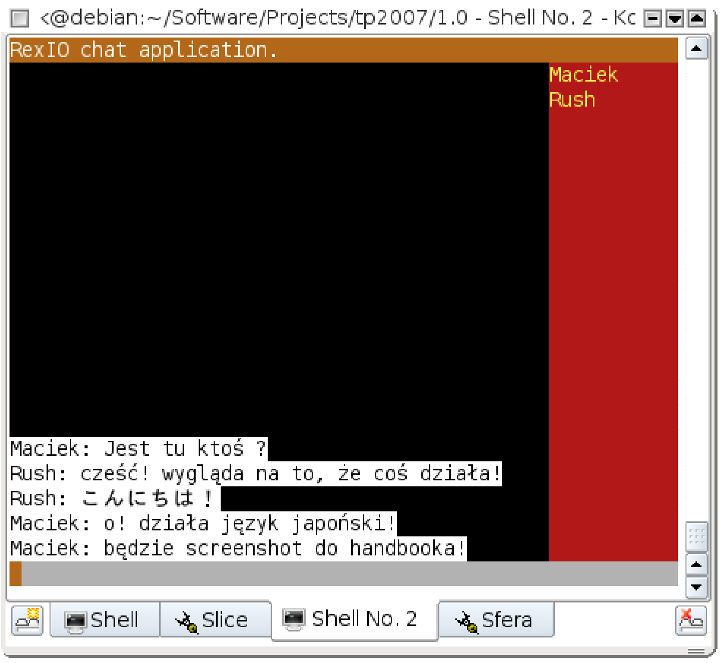
```
Witaj w aplikacji testowej RexIO

Demo to ukazaje działanie RexIO na przykładzie
sieciowej aplikacji do pogawędek.
Interfejs graficzny udostępniany jest przez
zwykły klient "telnet".

Port: 4555
Użyj telnet by połączyć się z powyższym portem.

                  ^   ^
         _________  ___
        / RexIO? \ (oo)\_________
        ---------- (__)\         )\/\
                        ||----w |
                        ||     ||



Miłego użytkowania!
```

Client welcome screen


&lt;@debian:~/Software/Projects/tp2007/1.0 - Shell No. 2 - Kc
Shell
Slice
Shell No. 2
Sfera

```
Witaj w aplikacji do pogawędek RexIO!

Provide your nickname and press Connect

  My login                              Connect


RexIO jest biblioteką kontroli interfejsu
tekstowego z wsparciem dla szerokiej gamy
terminali oraz sposobów łączenia. Terminale
zdalne jak i lokalne obsługiwane są przez
klasy o takim samym interfejsie.
                    ^   ^
         _________  ___
        / RexIO? \ (oo)\_________
        ---------- (__)\         )\/\
                        ||----w |
                        ||     ||
```

Client default chat screen



```
RexIO chat application.
                                                      Maciek
                                                      Rush









Maciek: Jest tu ktoś ?
Rush: cześć! wygląda na to, że coś działa!
Rush: こんにちは！
Maciek: o! działa język japoński!
Maciek: będzie screenshot do handbooka!
```

5 References

Following works are included in the library:

- __WHERE_AM_I__ macro was originally written by Curtis Krauskopf
- fileno_hack function was originally written by Richard B. Kreckel
- Scr::TI::Strings, Scr::TI::Numbers and Scr::TI::Booleans enums are based on macros in /usr/include/term.h file, by Zeyd M. Ben-Halim, Eric S. Raymond and Thomas E. Dickey.
- Scr::LocalScreen::TestForResize member function is based on comparable function in Berkley TELNET client.

Index