The monthly GNU column

Brave GNU World

Welcome to another issue of the Brave GNU World, which will be a little more technical this month.

BY GEORG C. F. GREVE

Even though some projects are probably of primary interest to developers, we still hope that less technical readers will also be able to draw new perspectives and be inspired by the projects shown here.

Twin

The first topic this month is Twin, [5] a multi-window, multi-application text-based environment by Massimiliano Ghilardi. Consequently, Twin is an acronym for "Text WINdows" or (even better) "a textmode window environment."

The project aims at people who seek to have an environment with several windows without needing or wanting all the features of X11 – especially its significant resource hunger.

Combining Twin with Links, a text-mode web browser, only requires about 5% of the resources compared to X11 with the graphical browser Konqueror. All applications that can be used on a console or in a terminal window can also be used under Twin.

In a time where graphic cards are trying to outrace each other with new features and where what was impossible to afford yesterday is available for pocket change the day after, this seems almost anachronistic. But as already explained for the RULE project in the last issue, [6] this is only true for a small part of mankind.

There is a group that profits from Twin, that is normally not the focus when thinking about new software: blind and visually impaired people. As they may depend on using Braille terminals, they have little use for graphical user interfaces.

With Twin, they can now also use a full environment with multiple windows and applications.

Technically speaking, the project consists of a server, called "twin" like the whole project. This accepts connections from the clients and creates or modifies windows according to their commands. Also the server dynamically manages the different displays and devices.

Twin currently handles the console with mouse support via gpm and every termcap or ncurses compatible terminal with mouse support through the "xterm" mouse protocol, if available. But it is also possible to use X11 by means of a simple X11 driver or the graphically enhanced gfx-driver for output as well as another Twin server on another machine.

The General Graphics Interface (GGI) is also supported, but since it still lacks keyboard support, this most probably makes it rather unsuitable for most applications.

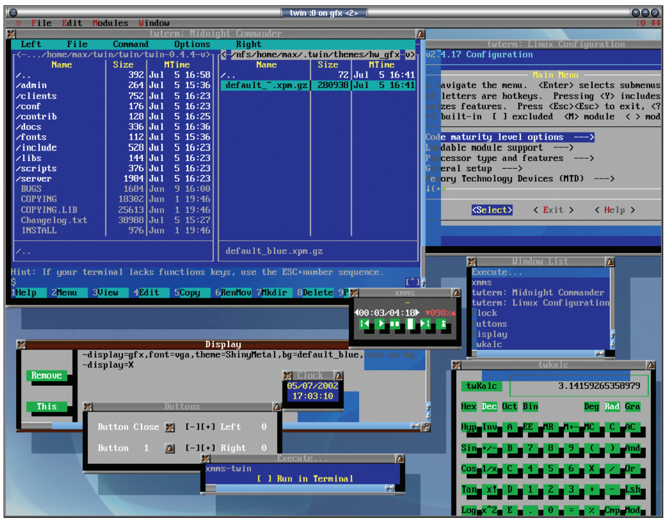
Among the other components are the libraries libTw, which handles the communication with the server, as well as libTT, which as the toolkit library provides an abstraction from of the more graphically oriented server-side functions to the more window/object oriented functions clients prefer. The third library is libTutf, a Unicode library, which allows transferring text from and to unicode. This library will probably become obsolete by using standard libraries some time in the future when the final open issues have been addressed.

Finally there are the clients. Currently there are only a few of them, of which two are built into the server. Both the window manager, which can be configured through a "~/.twinrc" configuration file, and a terminal emulating the console have been integrated into the server for technical reasons.

Other clients are an additional terminal emulator (twterm), a login manager similar to xdm/gdm/kdm (twdm), a system monitor (twsysmon), utilities to (de-)register displays with the server as well as other smaller clients that are more suited for testing than real work.

The project has been written entirely in C, one of the reasons for its small memory footprint – a Twin server usually requires less memory than the Bash shell. And of course Twin is Free Software – its licenses are the GNU General Public License (GPL) for server and clients and the GNU Lesser General Public License (LGPL) for the libraries.

Further development is pursued by Massimiliano in his free time, and he still has a lot of ideas. First he'd like to complete the toolkit library and its documentation, then more editors, task bars, file managers, web browsers, email programs and TTY based programs should be expanded to use it.


twin :0 on gfx <2>
File Edit Modules Window
twterm: Midnight Commander
Left File Command Options Right
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9P
twterm: Linux Configuration
<Select> < Exit > < Help >
xmms
00:03/04:18 090%
Display
Remove
This
Clock
Buttons
Button Close [-][+] Left 0
Button 1 [-][+] Right 0
Execute...
Window List
twkalc
twKalc
Hex Dec Oct Bin Deg Rad Gra
Hyp Inv A EE MR M+- MC C AC
Sin +/- B 7 8 9 ( ) And
Cos 1/x C 4 5 6 X / Or
Tan x! D 1 2 3 + - Lsh
Log x^2 E . 0 = % Cmp Mod

```
<-.../home/max/twin/twin-0.4.4-v>   <-/nfs/home/max/.twin/themes/hw_gfx-v>
 Name           Size   MTime            Name              Size   MTime
/..              392 Jul  5 16:58      /..                  72 Jul  5 16:41
/admin           264 Jul  5 15:36       default_~.xpm.gz  280938 Jul  5 16:41
/clients         752 Jul  5 16:23
/conf            176 Jul  5 16:23
/contrib         128 Jul  5 16:25
/docs            336 Jul  5 16:36
/fonts           112 Jul  5 15:36
/include         528 Jul  5 16:23
/libs            144 Jul  5 16:23
/scripts         376 Jul  5 16:23
/server         1984 Jul  5 16:23
 BUGS           1684 Jun  9 16:00
 COPYING       18302 Jun  1 19:46
 COPYING.LIB   25613 Jun  1 19:46
 Changelog.txt 38988 Jul  5 15:27
 INSTALL         976 Jun  1 19:46
/..                                    default_blue.xpm.gz
Hint: If your terminal lacks functions keys, use the ESC+number sequence.
$
```

```
v2.4.17 Configuration
                         Main Menu
s navigate the menu.  <Enter> selects submenus
ed letters are hotkeys.  Pressing <Y> includes
arizes features.  Press <Esc><Esc> to exit, <?
*] built-in  [ ] excluded  <M> module  < > mod

Code maturity level options  --->
Loadable module support  --->
Processor type and features  --->
General setup  --->
Memory Technology Devices (MTD)  --->
```

```
-display=gfx,font=vga,theme=ShinyMetal,bg=default_blue,root-as-bg
-display=X
```

```
05/07/2002
17:03:10
```

```
Execute...
xmms
twterm: Midnight Commander
twterm: Linux Configuration
Clock
Buttons
Display
twkalc
```

```
xmms-twin
     [ ] Run in Terminal
```

```
3.14159265358979
```

Figure 1: Twin showing off some graphical features in text

C++ Packages

The column continues with some projects that should make the life of C++ developers more easy and were all released by Christian Holm around December 1st 2002. [7]

When people communicate with computers, they need to agree upon a common language to be used in communication. Especially when this communication does not happen in real time or is very complex. That is for instance the case with programming languages or configuration files.

The basic problem is that functions for syntax checking and reading or evaluating of such languages can easily become very complex. When changes of definition or grammar become necessary, this often results in a very time consuming search for bugs.

For this reason, tools have been created that can automate the translation of definitions of grammar into functions that can read that grammar. Of course this means that the definition of grammar itself needs to be machine readable. The probably most common form of such a definition is the "Lookahead Left to Right Parsing" (LALR) context-free grammar.

One of the best and most-popular LALR(1) parser is Bison, [8] the Yacc equivalent of the GNU Project. Yacc itself stands for "Yet Another Compiler-Compiler" and Bison has deliberately been kept compatible to Yacc in order to ease transition from Yacc to Bison.

An application often working hand in hand with Bison is Flex, [9] which can be used to generate routines that allow dissecting a source input into single expressions, because it automates generation of source code for pattern matching.

Yacc/Lex--

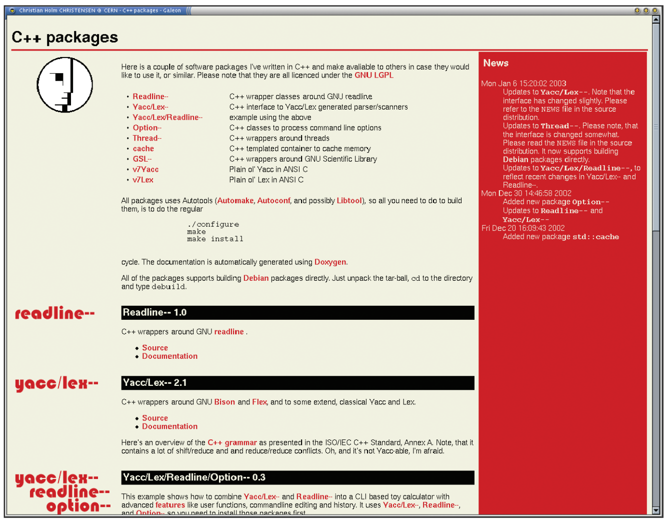
Both Bison [8] and Flex [9] usually create C source code. Is this code used in C++, they tend to clutter the global namespace; also there are no C++ interfaces available. For this reason Christian Holm Christiansen has written a group of header files called Yacc/Lexx--, which allow encapsulating the C output of Bison and Flex in C++ classes. The changes to the parser/scanner specifications were deliberately kept at a minimum to allow for greatest possible flexibility.

In fact Flex itself provides capabilities to generate C++ sourcecode, but the output was too inflexible for Christian's liking and also it didn't fit well with the parser classes generated by Bison. Therefore he wanted a common encapsulation for both.

Compared to projects like bison++, which has the advantage of direct C++ output, Christian sees the advantages of his method in being independent from the internals of the employed Yacc/Lex implementation. Therefore it is more stable with respect to changes in the Yacc/Lex projects and not immediately affected by their internal modifications.

But there are also some Yacc/Lex clones displaying odd behaviour and are not POSIX compliant; these can be problematic to use, which is a special problem of this project.

Christian plans to test more Yacc/Lex implementations and would be happy to receive help in this area.



Christian Holm CHRISTENSEN @ CERN - C++ packages - Galeon

C++ packages

Here is a couple of software packages I've written in C++ and make avaliable to others in case they would like to use it, or similar. Please note that they are all licenced under the GNU LGPL

• Readline--	C++ wrapper classes around GNU readline
• Yacc/Lex--	C++ interface to Yacc/Lex generated parser/scanners
• Yacc/Lex/Readline--	example using the above
• Option--	C++ classes to process command line options
• Thread--	C++ wrappers around threads
• cache	C++ templated container to cache memory
• GSL--	C++ wrappers around GNU Scientific Library
• v7Yacc	Plain ol' Yacc in ANSI C
• v7Lex	Plain ol' Lex in ANSI C

All packages uses Autotools (Automake, Autoconf, and possibly Libtool), so all you need to do to build them, is to do the regular

```
./configure
make
make install
```

cycle. The documentation is automatically generated using Doxygen.

All of the packages supports building Debian packages directly. Just unpack the tar-ball, cd to the directory and type debuild.

readline--

Readline-- 1.0

C++ wrappers around GNU readline .

- Source
- Documentation

yacc/lex--

Yacc/Lex-- 2.1

C++ wrappers around GNU Bison and Flex, and to some extend, classical Yacc and Lex.

- Source
- Documentation

Here's an overview of the C++ grammar as presented in the ISO/IEC C++ Standard, Annex A. Note, that it contains a lot of shift/reduce and and reduce/reduce conflicts. Oh, and it's not Yacc-able, I'm afraid.

yacc/lex-- readline-- option--

Yacc/Lex/Readline/Option-- 0.3

This example shows how to combine Yacc/Lex-- and Readline-- into a CLI based toy calculator with advanced features like user functions, commandline editing and history. It uses Yacc/Lex--, Readline--, and Option-- so you need to install those packages first.

News

Mon Jan 6 15:20:02 2003
Updates to Yacc/Lex--. Note that the interface has changed slightly. Please refer to the NEWS file in the source distribution.
Updates to Thread--. Please note, that the interface is changed somewhat. Please read the NEWS file in the source distribution. It now supports building Debian packages directly.
Updates to Yacc/Lex/Readline--, to reflect recent changes in Yacc/Lex-- and Readline--.
Mon Dec 30 14:46:58 2002
Added new package Option--
Updates to Readline-- and Yacc/Lex--
Fri Dec 20 16:09:43 2002
Added new package std::cache

Figure 2: Christians website showing his range of programs

Readline--

The GNU Readline Library [11] provides functions that allow integrating a versatile commandline into other projects.

Among the features of GNU Readline are a vi and EMACS mode, it can save old input, recreate it and allow editing it again or also complete the beginnings of previously entered commands similar to the csh shell.

The Readline-- project by Christian Holm Christensen allows C++ programmers to access the GNU Readline Library by means of C++ classes. Not surprisingly, C++ developers seeking to include a commandline interface in their applications are the main target group of this project.

The program originated when Christian himself needed a commandline interface to test his C++ parser, a task during which he also created the previous project.

The largest problem is that the library is not yet thread-safe, so it should be handled with care in complex applications. Fixing this and improving the interface are Christian's next plans for the project, because even though the interface is complete, he considers it unintuitive in some places.

Option--

With Option--, Christian provides a C++ parser for commandline options; a library that allows C++ programs to find and evaluate commandline options passed at program start.

The major advantage of the project compared with similar projects is that possible options are represented by template-classes, which makes the project very flexible. Option-- only works for non-positional arguments, though. So if the user needs to be forced by syntax to only use a certain option at a certain position in the commandline, Option-- is not a good choice.

Thread--

The last project by Christian Holm Christensen in this issue is Thread--, a project to use Threads in C++ programs.

Essentially, all computers work linear. If they get a task, they will complete it step by step in the given order with all their capacity. In most cases, this would only allow running one program at a time, however.

In order to allow working on several programs simultaneously - the so-called "multi-tasking" - the executing kernel of the computer, the processor, jumps from task to task. Each of these tasks in turn is again worked on in a strictly linear fashion, but the method allows splitting the computers capacity between programs.

As the programs and their tasks become more complex, working only strictly linear within a program is increasingly unsatisfactory. A solution to this problem is threading. Threads allow splitting programs into different "task threads" that again can be worked on linearly to solve different subunits of the complex problem.

The interaction and communication between these different threads of course also needs to be co-ordinated and controlled, a functionality which can be accessed by means of Thread-- from C++.

Different from similar projects like Boost::Thread, ZThread or Common C++, Thread-- does not distribute preprocessor macros throughout the source code. Implementation specific parts are instead put into Traits. This makes the library quite small and extensible.

Christian originally began working on Thread-- in order to test the thread-safety of Readline-- and according to him it works fine on GNU/Linux, but GCC versions of 2.95.x and below are problematic, so it is advisable to check for the GCC version.

The other problems are semaphores under Solaris and Threads under Win32; he could not test it on other platforms. Help with these problems as well as information about other platforms is very welcome.

TUX&GNU@school

Towards the end of this issue it is my pleasure to point readers to another remarkable column. Mario Fux, himself a long-time reader of the Brave GNU World, began last year to write a similar column dedicated specifically to Free Software in school.

By now he has finished 5 issues of the "TUX&GNU@school" column, which has found its new home on the FSF Europe home page. [17] I wish Mario, as well as Christian Selig and Kristian Rink, who support him as a kind of editorial board, all the best for the future.

6th EC Framework Programme

As mentioned [12], the FSF Europe [13] wrote a recommendation [14] to the European Commission on April 30th, 2002. The recommendation, first of all explained the advantages of Free Software for the region Europe and European countries in order to then suggest giving Free Software priority status.

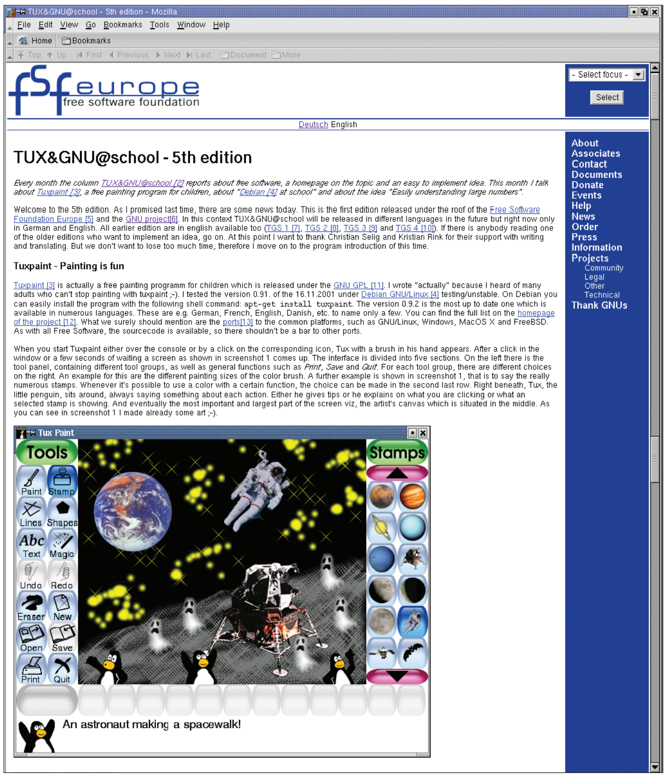
On December 17th 2002 the 6th Framework Programme was finalized and it seems that the recommendation to make Free Software the preferred form for project proposals has been heard. This effectively means that the whole budget of the IST Work Programme, containing 1725 million Euro, has been opened for Free Software. This is most likely the largest sum that was ever available for Free Software funding.

In order to now support companies, universities and research centers to launch projects for and with Free Software within this framework, the FSF Europe sent out a request [16] in which it asks all parties to get in touch.

Until next month

Enough Brave GNU World for this month, as usual I'd like to ask for questions, ideas and comments by mail. [1]

So much for now, until next month. ■



TUX&GNU@school - 5th edition

Every month the column TUX&GNU@school [2] reports about free software, a homepage on the topic and an easy to implement idea. This month I talk about Tuxpaint [3], a free painting program for children, about "Debian [4] at school" and about the idea "Easily understanding large numbers".

Welcome to the 5th edition. As I promised last time, there are some news today. This is the first edition released under the roof of the Free Software Foundation Europe [5] and the GNU project[6]. In this context TUX&GNU@school will be released in different languages in the future but right now only in German and English. All earlier edition are in english available too (TGS 1 [7], TGS 2 [8], TGS 3 [9] and TGS 4 [10]). If there is anybody reading one of the older editions who want to implement an idea, go on. At this point I want to thank Christian Selig and Kristian Rink for their support with writing and translating. But we don't want to lose too much time, therefore I move on to the program introduction of this time.

Tuxpaint - Painting is fun

Tuxpaint [3] is actually a free painting programm for children which is released under the GNU GPL [11]. I wrote "actually" because I heard of many adults who can't stop painting with tuxpaint ;-). I tested the version 0.91. of the 16.11.2001 under Debian GNU/Linux [4] testing/unstable. On Debian you can easily install the program with the following shell command: `apt-get install tuxpaint`. The version 0.9.2 is the most up to date one which is available in numerous languages. These are e.g. German, French, English, Danish, etc. to name only a few. You can find the full list on the homepage of the project [12]. What we surely should mention are the ports[13] to the common platforms, such as GNU/Linux, Windows, MacOS X and FreeBSD. As with all Free Software, the sourcecode is available, so there shouldn't be a bar to other ports.

When you start Tuxpaint either over the console or by a click on the corresponding icon, Tux with a brush in his hand appears. After a click in the window or a few seconds of waiting a screen as shown in screenshot 1 comes up. The interface is divided into five sections. On the left there is the tool panel, containing different tool groups, as well as general functions such as *Print*, *Save* and *Quit*. For each tool group, there are different choices on the right. An example for this are the different painting sizes of the color brush. A further example is shown in screenshot 1, that is to say the really numerous stamps. Whenever it's possible to use a color with a certain function, the choice can be made in the second last row. Right beneath, Tux, the little penguin, sits around, always saying something about each action. Either he gives tips or he explains on what you are clicking or what an selected stamp is showing. And eventually the most important and largest part of the screen viz, the artist's canvas which is situated in the middle. As you can see in screenshot 1 I made already some art ;-).

An astronaut making a spacewalk!

Figure 3: The latest version of TUX&GNU@School column

INFO

[1] Send ideas, comments and questions to Brave GNU World: *column@brave-gnu-world.org*

[2] Home page of the GNU Project: *http://www.gnu.org/*

[3] Home page of Georg's Brave GNU World: *http://brave-gnu-world.org*

[4] "We run GNU" initiative: *http://www.gnu.org/brave-gnu-world/rungnu/rungnu.en.html*

[5] Twin home page: *http://linuz.sns.it/~max/twin/*

[6] Brave GNU World, Issue #46: *http://brave-gnu-world.org/issue-46.en.html*

[7] C++ Packages: *http://cholm.home.cern.ch/cholm/misc*

[8] Bison home page: *http://www.gnu.org/software/bison/*

[9] Flex home page: *http://www.gnu.org/software/flex/*

[10] The Lex & Yacc page: *http://dinosaur.compilertools.net*

[11] GNU Readline library: *http://cnswww.cns.cwru.edu/php/chet/readline/rltop.html*

[12] Brave GNU World, Issue #40: *http://brave-gnu-world.org/issue-40.en.html*

[13] Free Software Foundation Europe: *http://fsfeurope.org*

[14] Recommendation by the FSF Europe for the 6th FP: *http://fsfeurope.org/documents/fp6/recommendation.html*

[15] 6th Framework Programme: *http://www.cordis.lu/fp6/*

[16] Call for participation of the FSF Europe: *http://mailman.fsfeurope.org/pipermail/press-release/2002q4/000047.html*

[17] TUX&GNU@school column: *http://fsfeurope.org/education/tgs/*