uSk Screenshot Download edx/EditX Screenshot
edx/EditX tgz

Twee Editors

Source code for small text editors

Feb. 29, 2004

*A 'twee' editor is one that is only a few multiples of the minimum size for a functional editor, without compression.

I have been a programmer since 1975, when I wrote my first Forth compiler, assembled it by hand into octal and then _hand toggled_ it into a binary register on a Bendix BDX-900 mini-computer. 1100 18bit words, if I recall correctly.

The second version of Forth was a real improvement. I hand assembled the opcodes into hex and hand keyed them in thru an alter/examine/execute hex monitor on an 8-bit 6502 microprocessor. Took about 1250 bytes.

The third time I had a line editor and assembler, both loaded from cassette tape onto the development system. There, I edited source from a keyboard and then assembled and loaded the resulting binary using the computer. How much better can it get?

What did I learn from the above? Forth, machine language and that I really, really needed a text editor. Have been writing, and re-writing them ever since.

What I learned in my early days of contract programming was that 60% of the project was the design phase, 20% was the entering code phase, 10% debugging and the rest maintenance. Proper design and debugging only required a line editor to get the requisite changes into the computer. So communicating with computers doesn't require much more than a command line that recognizes backspace and enter keys.

But then things got complicated. Modems became popular. Email and online accounts became geek requirements. Now I was communicating with peepul. And they require more flexibility in their communications. That was when the screen editor came into wide use. We programmers now had smiley faces for emotive connotations. The rest of the world sneered at us and moved on to 'word processors' with colors and fonts.

How many different ways are there to communicate between humans thru a computer? Just the factorial of the number of connections between humans using computers. Humans can and do change their mode of communication with each person they attempt to communicate with. The wide spread use of word processors now makes somewhat more sense to me.

So much for word processors. I was being paid to communicate with computers. Back to ASCII text editors.

As the pace for project development increased, the amount of time decreased. Management was paying to see results. So a more flexible editor was needed to demonstrate the progress on a project. Heh. What they got was screen editors that could fling data past them faster than they could read it. But volume was impressive and they were satisfied.

What does a GUI have to do with real-time OSs and hardware driver development? Not much. But GUIs were the new trippy toy and you could really dazzle people with the new editors. But I still keep ASCII screen editors around for when I really have to get the job done. Where do you think this software came from?

My first ASCII screen editor was Wordstar. Hated it, but the control key sequence came eventually. The next screen editor demanded I learn new controls, which I refused. Wrote my own text editor. With Wordstar control codes. Moved from CPM to DOS. Same thing. Learn new control sequence. Same response. Find source to already written text editor. Rewrite to Wordstar control sequence. (I will also mention there were a whole slew of Forth text editors I wrote, but that is for a historical perspecitve on Forth). Been doing one or the other ever since. Never professionally polished, just good enough to suit my needs. And it always teaches me something new and useful.

One final note. An absolute necessity for me, in a text editor, is an 'undo' feature. With that, I have both error recovery and a real time micro CVS while doing program development. If a text editor doesn't have undo, it is just a toy.

A Brief History of Interactive Text Editors

We'll skip the old punch cards and go straight to teletype.

In the beginning was the line editor. Open a file, list lines in a file, delete line(s) to kill buffer, insert immediate line/kill buffer to a specified point, write to file and exit. Undo in the form of a kill buffer. How much simpler can it get?

Then came the VDU (Video Display Unit, remember the ADM-3A?), a CRT serial terminal which enabled 'the screen editor'. Open file, display a full screen of text, move cursor, insert characters, mark blocks of text, copy/delete/save marked blocks, insert file, create file, save file, close file and exit. Not really that much more complex.

Then came the PC with mouse interface. Augment the keyboard text selection with mouse movement and button clicks. But this 'simple' change wasn't. It required a secondary task running in the background to unite asynchronous mouse motion and button clicks.

Then some genius recognized that the screen updating could be handled by the same background task that ran the mouse and, voila, the WIMP (Windowed Interface, Mouse Pointer) GUI (Graphical User Interface) was born. Display characters or paint pretty fonts (or pretty pictures for that matter), track point-and-click to select text or change/update the display.

After several years of shaking out, the multi-tasking versions settled into a common simplified form:

- •open a Window GC (Graphical Context)
- •set fonts and colors
- •open/create the text file for editing
- •editor main loop gets mouse and keyboard events, translates to insert/mark/delete/save/load/quit actions.
- •Each editor instance is treated as a single task.

If an OS main() executive is monitoring and providing the mouse/keyboard events, then it can intercept and pre-empt any other task in order to switch-to/kill any given task. Not quite the simplified logic of a line editor, now is it?

The following table of 'Twee' editors tracks this development cycle and provides the data for some observational conclusions. The group ordering is roughly that of the sequence that each genre was developed. Linux/UNIX is both near first and last because the UNIX command line started with line editors before DOS existed, but is late to the party for mature simple GUI text editors.

* Note: for the purposes of this discussion, the word 'functional' means: posseses a reasonable set of features to provide usable operation. The word 'operational' means: a less than acceptable set of features and is barely usable.

Legend:

size #bytes: Size of program in bytes. Forth lists the size of compiled editor code and size of kernel, including editor code.

name: Text editor name.

Source/Compiler: source type and specific compiler/assembler.

Engine: Text editor engine, i.e. supplied code or external library.

Undo: Has relatively unlimited undo/redo.

Max size: maximum edit file size. 'mem' means up to maximum available memory.

Interface:

line: insert/append/delete whole lines.

screen: direct cursor positioning thru keyboard control.

gui: keyboard + mouse.

Twee Editor Table

Size #bytes	name	Source/Compiler	Engine	Undo	Max size	Interface
		Forth Compiler/Iinterpreter				
588/9752	c4th.com	Forth c4th	Code	no	1k	screen
1984/9556	u4.com	Forth u4	Code	no	<40k	screen
4241/17395	4word.com	Forth 4word	Code	no	mem	screen
		Linux Command LIne				
6424	ue	C gcc	Code	yes	mem	screen
8472	ed	C gcc	Code	no	mem	line
11579	e3	asm NASM	Code	yes	mem	screen
15208	e3c	C gcc	Code	no	mem	screen
22168	mye	C gcc	Code	yes	mem	screen
		DOS Command Line				
2982	Ted.com	asm MASM	Code	no	60k	screen
3294	e2dos.com	asm TASM	Code	no	60k	screen
4570	ke.com	asm MASM	Code	no	60k	screen
4088	tm.com	asm MASM	Code	yes	60k	screen
		Win32 Command Line				
15864	e3.exe	asm NASM	Code	yes	mem	screen
19540	mye.exe	C LCC	Code	yes	mem	screen
		Win32 GUI				
7168	qikpad.exe	asm MASM	Lib	yes	32k	gui
8704	uRe.exe	asm MASM	Lib	yes	2M	gui
16896	uSk.exe	asm MASM	Lib	yes	2M	gui
101888	*Sknt.exe	C MSVC	Code	yes	mem	gui
		Linux GUI				
7032	u	C gcc	Lib	yes	mem	gui
9980	xedit	C gcc	Lib	yes	mem	gui
16600	uSk	C gcc	Lib	yes	mem	gui
31640	edx	C gcc	Code	yes	mem	gui
30488	ge	C gcc	Lib	no	mem	gui

* for reference purposes only. Not available for download.
Note: 'u' in the name of an editor is pronounced 'micro' not 'you'.

---

forth compiler/interpreter:

Go ahead, say "Lisp based emacs is simple." Make. My. Day.

Which is why Forth is included. It predates UNIX, and is famed, nay, notorious, for it's simple editors.

See the c4th.com screen text editor for a simple 1k source, 544 bytes of code screen text editor. Of course it is limited to editing only a few k of text, but it shows how small an operational screen editor can be. Oh, sure, be strict and include the interactive, open-ended, programmable, incremental compiler/interpreter with file I/O and it is still only 9752 bytes in length.

The u4.com forth text editor shares the same source with the 4word text editor. The editor code sizes are 1984 and 4241 bytes. u4 is a 16-bit version of Forth, while 4word is a 32-bit version that gives you access to the entire 386+ physical memory. What a difference an architecture makes! The Forth kernel sizes are 9556 and 17395 bytes, respectively. (What can I say? 32-bit code is larger). The u4/4word text editors are basic screen editors with forward search. Functional for Forth programming purposes. Operational for a generic text editor.

Linux command line:

Maybe I finally got it right. ue is my take on the optimal mixture of code size and functions. String search, undo and tab character handling in a 6.4k screen editor. I have looked for 20 years for a basic ASCII screen text editor. Finally quit looking and converted Anthony Howe's 1991 Obfuscated C Code winning editor, ant, to a basic screen editor. Good for rescue disks and roll-your-own projects.

Ed is the classical line editor. 8472 bytes. Everything you ever _needed_ in a functional text editor.

Albrecht Kleine's e3 screen editors are about the finest examples of maximum utility/artistry in a small package. Simple, e3 aint. The code optimization tricks in e3.asm can keep me up all night (they have). But it is the smallest (12550 bytes), most complete standalone screen text editor, with undo, you can find for linux. Extreme functional artistry.

For the binary or OS challenged, Albrecht also provides e3c.c. File size, there, is 15208 bytes. Multiple interface styles including Wordstar. Extremely functional.

Mye is an updated version of Yijun Ding's ee.c. It utilizes termcap I/O and also compiles under Win32. Yijun's original code simplicity is reflected in a final binary size of 22168 bytes. Has nearly identical capacities to Edx, including undo/redo. Basic WordStar control key clone. Functional.

DOS command line:

DOS provided the ubiquitous environment that produced some of the simplest and/or smallest screen text editors ever.

See Tom Kihlken's ted.asm for the results of a collision between a line editor and cursor/function keys. And in only 2982 bytes of assembler. Now that is elegant programming in a functional editor.

Albrecht Kleine has e2dos.com. Artistry in a WordStar clone at 3294 bytes. Quite functional. (Albrecht says e2dos.asm is TASM source, but I was able to compile it with MASM, your choice).

The ke.com editor is my hacked version of ted that uses the WordStar control sequence, with the addition of a search function and status line. Size blossoms to 4570 bytes. Functional.

But Brian Kelly's tm.com editor is the icing and the cake. A partial list of features include 4088 bytes in size, command line loading, incremental forward/reverse search and undo. Byte for byte the most efficient end-user application I have ever seen. And his code optimizations are a full course in assembler language programming. Beyond functional. A dedicated work of art.

Win32 command line:

The Win32 command line suffers from a lack of text editors from the Windows world. (Cygwin may be in the Windows world, but it is not of it.) Same source for e3 and mye, but in a Win32 environment. See above for functional descriptions/comments. e3 still rocks.

Win32 GUI:

The Win32 GUI world suffers from bloat. "What's the problem? It takes up a smaller percentage on the new, bigger hard disks." A quote from an anonymous MicroSerf when I asked why new apps were always larger.

Presented are three WIMP GUI editors written in assembler to buck the bloated trend. All include the standard CUA (Common User Architecture) cut/copy/paste with either keyboard or mouse.

All three editors are derived from the MASM32 assembler kit from Hutch and Iczelion. These guys are walking lessons in Win32 assembler, as well as superb programmers. Worth studying in detail.

MASM32 is the definitive package for Win32 assembler programming, including MASM 6.1x, supporting binaries, include files and libs, with tutorials and excellent working examples. (Don't talk to me about the bloated Cygwin and NASM doesn't include all the extras.)

Qikpad is an example asm wrapper derived from the MASM32 kit around the Richedit text control, limited to edit file size of 32k, max. But it does include undo. It shows how low you can go and still be functional.

uRe (u Rich edit) is an improved version of Qikpad that opens files up to 2Meg in size. Search and command line file loading were also added for a binary size of 8704 bytes. Functional.

uSk is a full blown notepad lookalike, with toolbars and everything. The pretty eye candy and added functions bring the size up to 16896 bytes. Contrast that with Win Notepad, similar functionality, but 50960 bytes in size. Very functional.

SKNT (SideKlik for NT) is my 10 year old work horse for Windows. I purchased the source for it back in 1991 and have hacked it beyond recognizition. Sorry, but I cannot make the source available. I bought the source, not the copyright. Included in the table for reference only.

SKNT is a functional clone of all the features that so endeared me to Borland's 1984/85 Sidekick TSR editor. The best all around application ever written for a computer. Borland's SideKick had a popup text editor with a 48k max file size, calculator, phone book/dialer, calendar and ASCII char/hex conversion table. All in a 39k com file. Actually made a $5000 Compaq paperweight usable. You might also now understand the tribute I pay with some of my text editor names.

But SKNT is a much better editor than Sidekick. with toolbar, WIMP GUI, multi-windows, file size limited by available memory and undo/redo. Has a builtin run command, support for grep in specified directories and external calls to calc, calendar and Win3x phone dialer. Written in C, all in 101888 bytes. And this isn't a wrapper around a richedit text control. Not much in the windows world that comes close. This does provide a baseline for a nominal size of a complete Win32 GUI C source text editor.

Linux GUI:

Lets get the worst over first. Xedit. Mutant sport of emacs. Wrapper around the Xaw textWidget.

First impression: "Of course it's hard to use. It was hard to write!". Which was pretty much the case until Xfree86 4.xx came along and added undo and position_changed callback. Now you can know where you are when you make mistakes and undo them. Of course you have to enable undo with the app-defaults file.

Which is the savior of xedit. By editing Xedit.ad and then copying it to

'/etc/X11/app-defaults/Xedit' (or the equivalent on your system)

xedit becomes something rare: a functional, stable X text editor. In only 9980 bytes!

No, no, not the bloated Xfree86 4.xx xedit (don' need no steenking spell-checker). Use the source from Xfree86 3.3.2 and add the following line to the xedit apps-default file

'xedit.textSource*enableUndo: True'

You may also want to change the key bindings. After spending 6-10 hours of editing xedit's app-defaults file, Xedit, you might end up with something useful. And along the way learn that you can also _add_ functions by compositing multiple key functions onto one key. See the supplied uSk.ad and U.ad for working examples of Wordstar style remapping and multiple function compositing.

A minimal operational text editor app can be constructed. For file loading, it inserts a named file into an empty file. Text editing is the same that is available from xedit. But the only way to save the text is to select the whole file with a mouse and paste it to a command line of 'cat > file.txt'. Terminate with a '^C' to the command line. Sorta defeats the purpose of a text editor. But it only takes 4170 bytes. Left as an exercise for the reader to contruct from the u.c source (hint: remove everything that is not required to create the Xaw window and associate it with the textwidget).

u is a minimal functional wrapper for the Xaw textwidget. Loads files from the command line, creates a new empty textwidget, inserts files at the cursor, saves file, does forward/reverse search and replace and undo. In 7032 bytes. You can do everything from the mouse/keyboard, so why do you need menus or a toolbar?

For your geek mother, there is uSk. A notepad clone, Xaw style. Super-set of 'u', but with a status line, menus and toolbar. Provides help by invoking usk on '/etc/X11/app-defaults/uSk',
has an about dialog, opens a second usk, calls rxvt and allows the source_changed flag to be reset. Also has goto line#, a decent X file picker, dialog and messagbox support. As functional as any X text editor I have seen. The resulting size is about 16600 bytes.

Funny, thats almost an identical size to the same functionality available from uSk.exe under Win32 GUI. uSk on X, is written in C, uSk.exe is written in assembler. What's wrong with this picture? Answer: uSk.exe has to generate the Search dialog box for the Richedt20.dll. libXaw.so.7 has a Search/Replace dialog as a builtin. Interactive dialog boxes are very expensive in setup and creation of the individual controls/widgets, regardless if it is Win32 or X.

Edx is an on going proof of concept. It takes and converts the ASCII terminal text engine from mye and runs it under a bare X window. Edx provides all of the functions operational under terminal mode. Edx is now feature complete with full mouse and selection integration. Edx is functionally equivalent to SKNT, minus the menus/toolbar and multi-window capability. Size is a very reasonable 32856 bytes for a complete text editor under X. WS clone. Very functional.

Provided for comparison, is ge, a near functional GTK notepad clone which uses the GTK textwidget library. Much less functional than notepad, but in 30488 bytes. Not even close to Xaw in features.

Conclusions and Projections:

The absolute minimum size for an operational ASCII editor is about 1k, including File I/O, on a 16-bit processor. For a functional screen editor, the empirical minimum size seems to be about 3k. For a professional grade editor, 4k.

Go to a 32-bit command line and the minimum size blossoms to at least 10k. The larger byte count seems to be in the operand size and the OS interface overhead. Could also be in just plain extra functionality. That is, if you are writing in assembler.

Code size goes up by a minimum factor of 1.5 or larger when you switch to C code.

With GUI you get text (widgets and controls). Makes for small binary size, but depends upon resident dynamically linked libraries. The problem with that is, the functions provided may not match your needs, and the API to integrate your needs is only available at the library source level, which may not be available. I.E. the API dont exist.

The Win32 Richedt20.dll is 431k, while the entire libXaw.so.7 is only 335k, of which the textwidget is approx. 30% or 102k. To put it into perspective, a text editor (15k), which is just a wrapper around a text widget or a text control (100k+), come to a total size of 120k+. Or write your own editor, in C, custom fit to your needs for about 100k...or less.

Edx (aka EditX) is a basic text editor construction kit for bare X, which comes in at around 30k. See above. It is a feature complete standalone pure X text editor. No menus or toolbar, just control and function keys plus mouse. Undo/redo in only 1120 bytes. Dynamic buffer allocation in around 1100 bytes. Usable cursor mouse integration in about 2k, vertical srollbar and 3D effects in 1.3k and X selection in 2k. The logical breakdown of components seperate quite nicely into the X interface in edx.c, while the editor engine is complete in eeng.c. Code kept as simple as possible, but no simpler.

For assembler, you have a minimum size of the edit engine being about 12.5k. Add the minimum overhead to XtCreateManagedWidget (non-functional u size of 4k times the . 66 scale factor for assembler) of 2.7k plus 1.7k for menus (.66 times 2.5k) for a grand

total of about 16k to port e3.asm to X. This would be editor + window code + menus+undo.

If you get/have a complete text editor+source for any of the above platforms, please let me know. I will include a link from the Twee Editors table.

Back to Modest Proposals