Why I have gone the GNU route

Alan Robert Clark

November 2, 2004

1 Introduction

I am often asked what my problem is. *I* don't know—how should *I* know :-). Fact is, many people are puzzled by my adherence (as far as is possible), to GNU principles and products, and my aversion to all things Bill. These are *my personal* opinions as to why the standard executable-only software (please buy the upgrade) just doesn't cut it for me.

2 What is GNU?

Gnu's Not Unix. Its official, and consistent with the standard recursive acronyms that abound in the unix world!

Its about freedom. Of course, GNU is free, and it isn't free, if you see what I mean.

Many people confuse GNU software with Freeware, or shareware, or anythingelseWare, that is so popular in the DOS world. The GNU licence effectively ensures that the CODE, or SOURCE is free. You may pay to get the executable, or some support, but the code is *available*. Having said that, most GNU products are free in the cost sense too.

There is also confusion between GNU-project products and products that are distributed in terms of the GNU License. The GNU project as such can be found at "www.gnu.org" and has many quality products. Many other people contribute software, to be found all over the show, of excellent to shoddy quality, licensed under the GNU "GPL" or CopyLeft.

Many people (particulalry non-programmers) are puzzled by the insistance on the availablility of the code—it is important for the future growth of the product. Not only are *you* able to tinker with the software, but so can others when the current maintainer gets run over by a bus, gets too busy, or simply gets tired of it.

Some of the GNU software that I use has gone through several "maintainers" (PCB, DOSemu, MusicTeX), and has gone from strength to strength! Another advantage is that you have an international team of coders actively working on the software. Most have never met each other, but all contribute code, or patches, or maintain sub-parts of the project.

Many think that the above model would simply produce nightmare code! There are some good guidelines to follow, however, and the exposure of the code to others generally *tightens* it up! Peer-reviewed code is far less likely to contain bloopers that you simply can't see because you wrote it! GNU software is by no means bug-free, but it crashes a heck of a lot less than Bill, and the bugfix is *very fast*, without a wait for the next 6-monthly "service pak" that is almost as big as the original software (and costs as much)!

3 OK, but *really* Why?

We'll start with document preparation, a never-ending task in an engineer's life. By *document*, I don't mean those things that Turd thinks are documents, I mean 100 page reports with hundreds of figures, references, sections etc. A quick tour of the history follows:

3.1 WordStar

The late, great classic, useless at maths, pictures, but wonderful :-) One of the only editors for YEARS thereafter with *column* cut-and-paste!

3.2 Manuscript

Manuscript was a brilliant wordprocessor created by Lotus. It was truly the Greatest thing since sliced bread in about 1988. It handled postscript graphics (well), did equations really well (via a similar syntax to TeX), and had text and preview modes, which meant that it was really fast for text entry (really important in 1988!!). It printed as it previewed. *NO* little quirks or "funnies". What you previewed is what you got! The Manuscript upgrade to 2.1 was a really good one. It used a "folding editor" where you could hide sections, subsections etc. This was a really brilliant feature, which I have not yet seen efficiently replicated, although many folding editors are out there. It also encouraged the LaTeX-like structure in documents.

But there were no references, figure refs, cross refs, etc. These had to be done by hand. Hence we all used the harvard system, and at the very last moment, we did a search-and-replace to numbers! Still used by people in the department. One problem was that it was a tad unstable on large files. A colleague's PhD was 600k or so, and every now and then, it would corrupt it whilst saving. Back to the previous version!

This was my first introduction to the disastrous nature of Proprietary file formats (ie non-ascii), which are almost impossible to repair after they have gone awry.

Lotus unceremoniously dumped Manuscript, and graciously allowed you to buy AmiPro as a replacement.

3.3 AmiPro

The "successor" to Manuscript, bought into Lotus from the competition. It had no Postscript capability, no math capability, and the "Manuscript Document Import" facility converted text. Only text. A later version had a TEX maths engine, but could not do *inline* equations!. Well, it could, but for heaven's sake don't move the paragraph! The text of the paragraph moved, but the inline equation stayed put. I discovered this whilst setting an exam: I had a question which said something about a 50Ω transmission line. I then moved the question in front of another one, and it left the Ω sign behind!

This was my first introduction to the disastrous nature of Proprietary maintainers, who could sell, buy, or drop a package, regardless of the user's need. The summary dropping of an excellent package really cheesed me off. It was to be repeated later in other "acquisitions" and "mergers". The replacement of that package by a grossly inferior product (from an engineer's point of view, I guess most business types were most happy) cheesed me off to an exceptionally large extent! They could have released the source code!

Again, engineers do not do (or should not do :-) office memos in one-of-twenty-million fancy formats, that occupies two lines and 40 megs, a *document* contains maths, figures, cross-references, citations. . .

Another thing that escapes these types is that a document also needs to be *maintained*, as well as the software etc, hence an awful amount of time is spent massaging older documents (and esp graphics) into newer document packages!

3.4 Turd

The classic . . . Proprietary file format. . . change (still called .doc!) infuriated more people than MS thought it would. It was a blatant "I will force you to upgrade to remain compatible with your cleverer friends" tactic. It certainly blew the "If its Microsoft, its compatible" line out the water. Some of my colleagues are still of the opinion that all MS products are compatible though! (Sheesh, never mind the products, what about the product *versions*) Turd still doesn't handle postscript at all well (you get a Bounding Box if you're lucky, if not, it doesn't work—eg Matlab output (which is Blue Book correct postscript)). The mathematics is absolutely horrendous—it looks hideous at the best of times). Does not have a citations database interface (ie References) ala BiBTEX, and even its cross-references are weak.

The biggest problem with Turd is the way people are encouraged to "fiddle". You get headings in wierd unreadable fonts, with no guidance as the the heading "level" etc. (I recently received a press release in Brushscript, at 10 points. Nobody in the office could actually read it)

4 What Else?

Graphics packages:

4.1 TurboCAD

Excellent little package—rather a torturous route of exporting and etc to get figs into Manuscript etc, but it worked well, and was dumped. (There was a much later, revised edition after yet another aquisition, but it was essentially dumped.

4.2 Freelance

Did reasonable drawings, in a completely proprietary standard! Worked well with manuscript though, via a metafile output that did not always get everything right. Altogether, one got used to the idiosyncracies and actually had a working system, although a bit tedious at times..

It was dumped.

4.3 No real successor

There has been no real successor to these graphics packages, with various ones being tried, but unless you go for a mega-buck package like AutoCAD, there is not much to touch xfig (naturally not available under MsLoss :-). These days, I am a convert of GNU pic, using the GNU m4 based "circuit macros" as a basis for my graphics. It *really* works exceptionally well.

5 Compilers, editors and etc...

Yet another tale:

5.1 Turbo Pascal

I loved Version 3 :-) (limited by 64k size etc, marvellous!). I stuck with it (ie paid for 6-monthly "upgrades" aka bug-fixes) until version 7.0 (ie through about 7 upgrades—various sub-point versions too) Notably, even by that stage, although it could handle DPMI, and tons of memory, variables could *still* not be bigger than 64k!

I would hate to add up the cost of the 6-monthly upgrades... Borland brought out a brilliant text-based windowing Toolkit—Turbo Vision. Borland dumped it! The disastrous nature of Proprietary maintainers again.

And there we were—stuck with the TV bugs, and completely held to ransom! Note again, *COST WAS NOT THE ISSUE*, I had reams of software written with this toolkit, which was a binary release, with bugs. Eventually they also brought it out in Borland C++, and then released the source code! it is still a popular Toolkit, and has been ported to many platforms including Linux.

5.2 Brief

Great little programmers editor, many good features, inc column or block-wise editing. Not of tremendous use under Linux, though. Brief was by a company called UnderWare. They sold out to Borland, who released it as their main programmers editor for *one upgrade only*, thereafter going the WYSIWYG route too. Brief basically died after that, no updates etc.

5.3 Crisp

Crisp was the *shareware* version of brief, compilable under Linux. Great! used it for several years, but no colour-syntax highlighting etc. Crisp moved over to being real-money ware, but the difficulty was the source. 2.2e was such legacy code that it wasn't really worth the effort in doing anything about it. The commercial Crisp is now available under Linux, NT, etc, but a friend has it, and its Buggy. It also has developed the attitude of: Gee Sir, yes, that it a bug, thanks for bringing that to our attention; here: buy an upgrade to fix it.

Again, GNU is not Freeware or Shareware—I was still held to ransom.

5.4 Borland/Watcom/MSVC

Ever tried taking a "bog-standard" Borland-C++ app to Watcom or to Visual? Let alone the library inconsistencies, how about the differences in longevity of i in "for int i="??? The only (compatible) way around this is to declare "i" at a higher scope, ala Pascal, and go with the "for i=" syntax.

The real problem is that you can wait for months before a bug-fix *for which you then have to pay!* The most classic response ever, was received by us when we used M$ SourceSafe as opposed to RCS/CVS revision-control software: "What you are wanting to do is exactly what we intend SourceSafe to be used for. Unfortunately the inclusion of sub-version numbers will be a big performance-hit in the software at present. This will not be solved in the next release of SourceSafe. Thank for choosing M$ SourceSafe." The last sentence killed me. I now use CVS, and my colleagues in the team that run '95/8/NT use CVS-RCS, a shareware package that interacts perfectly with my Linux Samba-shared drive.

12 June 2000. So you thought it was all over, that Bill had finally triumphed? Well, we used to use PVM (Parallel Virtual Machine) for parallelizing SuperNEC computation, but now PVM has largely been surpassed by MPI(Multiple Processor Interface). Most apps used M$ VC++ and M$ Fortran4.0 THEY DUMPED IT. Have to fork out for Digital's Visual Fortran. Once again etc etc.

5.5 O/S2

Before 3.1 had a TCP/IP stack of any note, I switched to OS/2 version 2.0, for 800 bucks. The netware requester weighed in at 600 bucks and the TCP/IP at about the same. Only found out *after that* that they did not work together!!!!

This was later fixed, but not marvellously. Dos support wasn't all that hot either. I lasted about two months before switching to Linux in October 1992. (Remember, 1990 bucks were actually worth something!!!)

The oldest file that I still have in my home tree is a set of m-files that I used in matlab 3.5 and is dated 29 December 1992 :-)

5.6 Toolkits

Originally I used Prof. Walkers' Screen Management Library under TP3, migrated it to graphics under TP4, added all sorts of extensions under TP5, and then moved to TV. No sooner do I embrace Turbo Vision under pascal, which eventually gets brought out under C++ (With source, nogal) than it gets dropped. OWL is the way forward, which gets dropped (in essence, anyway), then Macrosloth's Foundation Classes come along in their various *incompatible* incarnations...

A man can get tired of reworking code. Some of my apps (”wkhm”) are still around and useful, but there are executables under TP4 (with the SML), TP5 (with TV) TP6 (with OOP)(Actually also TP5.5) & 7 (with XMS, non 640k limit management), and GNU-Linux! I have tried GNU-DJGPP(Dos), but the lack of support (at porting time) of standard Un*x utilities was a problem. In essence, no extra functionality was added in each of these versions!! its just plain reworking.

6 So?

If I would add up the amount of porting and re-porting of documents, code, toolkits that I have done... If I would add up the cost associated with this... and the cost of the "upgrades" of the packages themselves... In a word, it is "fed-up". I have been around for too long in the software world. I am sick-and-tired of being held to ransom, and of being dictated to.

Again, though, the major problem is the lack of source. I once designed an editor "Ted" based on the Turbo-Vision editor example code (buggy as hell). Fixed the bugs, and added wordwrap features and autosave that the original did not have. A popular DOS UseNet news-reader (Trumpet) was based on the same original code. But I couldn't get autosave implemented in the news-reader as although it was free of charge, it was not free in the source sense. Quite a number of emails passed between me and the author of the reader, but *he would not agree on my method of implementing the features.* So the Trumpet had no autosave, and no word-wrap. (At that stage, for some reason, we were having regular power cuts).

This illustrates again that if I had had the source, I could implement my own version that the author doesn't agree with, but that I want!

I have been forced to massage countless programs and documents into some other form, simply because of an upgrade, which usually has some "must-have" feature that is the attraction in the first place. I suppose I could still use

Manuscript, but I write longer documents now, with more and better-looking graphics, and have more than 640k in my machine :-)

7 What I do use

Hence my attraction to code that is *freely* available, *maintainable*, not subject to “buy-outs” and political decisions etc etc.

7.1 Editor

As an editor, VIM; in *everything.* Its extensible, and runs on any platform known to man...

I use it for composing mail, writing C++, html, Microchip Assembler, LaTeX documents. It is free, strongly supported by a large user community, and a large contributor community, has many many Web pages devoted to it, is largely Vi friendly, has column/block operations, syntax-highlighting, language-specific commands, ie a key binding to ‘make’ calls ”make” in c++, ”latex” for LaTeX, etc. It compiles on every platform known to mankind including Linux, all unices, NT, 95/8 Dos, OS/2, Mac, Amiga......

Its extensible, remappable, programmable, compatible with a large range of “usual” Un*x stuff.

My customizations of it can be found here.

7.2 Document typesetter

As a document typesetter, LaTeX. Its extensible, and runs on any platform known to man...

It was *designed* for mathematics, has a decent *structure* at its heart, has good olde fashioned ASCII as its file format. No matter what, you can always retrieve most of an ASCII file, even if it has been corrupted!!!!

Has decent html tools (TtH, latex2html), has decent self documentation tools for packages (DocTeX, aka .dtx), handles extremely large documents without a hiccup, has intelligent multipart document support, uses any decent editor of your choice, has exceptional postscript support (dvips), excellent and entirely accurate previewing (xdvi, or yap on the MsLoss world (MikTeX)), has extensive support structures ”comp.text.tex”, and the CTAN—Comprehensive TeX Archive Network, a collection of ftp sites with all sorts of (free) add-on packages, there is a local mirror.

For my figures, I use ”xfig”, a truly excellent package which allows me to embed LaTeX maths etc in my figs. I use the ”pstex” output method, and thence ”dvips” to get truly good quality figs.

For circuit diagrams, and increasingly for any graphics, I use a truly excellent text-based bunch of macros “circuit macros”, some of my figs can be found here.

7.3 Compiler

As a Compiler, G++, the GNU compiler. G++ runs on any platform known to man... If I need pascal, p2c does a good job, for fortran, f2c does likewise, although there is a "native g77" that I now use in preference.

Decent tools such as debuggers (xxgdb), segfault and memory hole traps (Electric Fence), decent Make (GNU Make), auto code-writing thingy's (yacc and bison). IDE's also exist (which I dont like or use, I use VIM) such as wxpe, a Borland Turbo-Vision look-alike, etc. In addition to this, we have benchmarked G++ code versus various other compilers (on the same machine) and have had better results, sometimes stunningly better results.

7.4 Utilities

As a bunch of utilities, including a shell, the GNU tools. These have been ported to every known platform that g++ compiles on, including bash for NT!!! This is why the system I run is actually GNU/Linux. Most of the utility and system software is GNU, and the Kernel is Linux. GNU/FreeBSD also exists, as well as GNU/NT etc etc.

7.5 Electronics Stuff

I use the *REAL* SPICE, sans restriction on nodes etc that the commercial chaps slap on their version which run exactly the same code! I use PCB by Thomas Nau to design my printed circuit boards, and the Circuit macros by Dwight Aplevich for schematics. There is a wealth of stuff out there! (try octave as a MATLAB replacement).

7.6 Toolkits

As a windowing toolkit, I am going to investigate gtk+, the Gimp Toolkit. A native 'Doze port is available. ie Code should be portable across platforms. In the past I have used Turbo Vision, which is available for '95/8 and Linux, but I find a lesser need for a GUI-like front-end these days!

Most of the serious technical windowing code I am involved with is written in MATLAB, namely SUPERNEC, and Visual CASED, an Electromagnetic Method-of-Moments and a Variable Speed Drive State-Space code respectively. Rumour has it that The mathworks also releases a 9x/NT port of MATLAB, so code written with this toolkit is platform independant too :-) :-)

The online version of this document is "http://ytdp.ee.wits.ac.za/WhyGNU.html"