

Re: Anybody here endure C/Cpp? (.h to .inc conversion)

Source: <http://coding.derkeiler.com/Archive/Assembler/alt.lang.asm/2005-01/0120.html>

From: Beth (*BethStone21_at_hotmail.NOSPICEDHAM.com*)

Date: 01/06/05

Date: Thu, 06 Jan 2005 01:00:16 GMT

Phil Carmody wrote:

> *Beth* wrote:

> > *Oh, yeah, I'm not saying X is perfect...but compared to Windows'*
insanely

> > *wasteful and pointless (and illogical) design, X is an "angel" of GUI*
> > *design...*

>

> *Tha absurd thing is that someone in the past once said:*

>

> *"Programming X-Windows is like trying to find the square*
> *root of pi using roman numerals. "*

>

> *When _that_ has suddenly become the more elegant system, then something*
> *somewhere else has gone horribly, horribly wrong. (I quite like Xlib*
> *to be honest, it's trivial to write a C++ wrapper around it that is*
> *wonderfully easy and natural to use. Not particularly useful for real*
> *world programs but fine for simple monitors (e.g. xload) or graphics*
> *demos.)*

No, this is a confusion between "standard / design" and `_implementation_...`

This is like blaming all car crashes on Ford...or blaming 9/11 on Boeing...

`_X_` is just fine...something like XFree86, on the other hand...now, there's a different matter altogether...

I popped your quote into Google and found an "anti-X" webpage that was making loads of complaints and, for example, it was mentioning that some X "clock" application takes 1.4 MB and another takes 600KB...

Again, are they blaming the correct thing? Because I wrote a simple X program here using C and it comes out at around 12KB...indeed, this seemed rather large for me, then I noticed I had a reference to "printf" (I was using "stdio.h" and "printf" solely to print out a "sorry, could not connect to X server!" error message...just to be "polite" when things go pear-shaped or someone tries running this X program when X isn't running

:)...so, I created a classic K&R "hello, world!" program to see...and, sure enough, the "hello, world!" comes out at around 11KB!! Okay, kick out the "printf" from the simple X program and...yeah, we get around 11KB...

>*From the looks of things, the majority of these programs is some bloated GCC "startup" code...because the X & libc comes out at 12KB, the libc alone comes out at 11KB, the X alone comes out at 12KB...and I'm rounding the figures there so it's actually bytes, not KB, between them...not quite to the highest Descartean levels of scientific proof but what else could be to blame for the 12KB file size here? And I've seen executable ELF files that are only a KB or so (you can make it mere bytes, if you get really "extreme" ;)...and one assumes LD won't link in what isn't used by the program...the only other common thing I can think of between these programs that could be taken about 11KB is the "startup" code from GCC (you know, handling the "argc", "argv" stuff and calling "main" :)...)*

But perhaps they mean RAM (as the complaint was "vague" about what "takes 1.4MB" meant...RAM? Disk space?)...on this score, if you write a simple Windows program and look into "Task Manager", every program is reserved 1MB of RAM before you've done anything at all, if it does anything "GUI"...what's that all about?

This is just poor implementation on both scores...there is no particular reason why either should take the absurd amounts of RAM that they do...I mean, there's been GUIs for an 8-bit 64KB Commodore 64...my old Atari ST only had 512KB (and the GUI did NOT take up the majority of it or anything at all)...I've written my own (very basic) GUI in BASIC, which was in real mode, so I just couldn't take 640KB just for the GUI or there'd be nothing left for anything else...

I mean, what exactly takes that much RAM? BOTH these GUIs don't have "persistent" window displays (that is, it does not "remember" what's in a window in any way...when a window gets covered and then "exposed", the application has to "repaint" the missing contents, as the GUI is NOT in any way keeping track of what's in the window at all...that's all dumped on the application...which is a terrible GUI design too that makes GUI programming unnecessarily complex...but, well, they are ALL like that, unfortunately)...so, the RAM isn't going on "backing store" (although, in X, you can ASK for a "backing store" to do just that...perhaps all these programs are abusing this facility? In Windows, you can do something similar but not quite the same in asking Windows to store what's underneath a window in a bitmap (and that's IF it's got the RAM or can be bothered to do it...it's a "hint", not a "guarantee" that Windows does it :)...supposed to be used for small windows like menus and things that don't take much RAM to copy the contents underneath to save making all those "repainting" calls...although, as Windows is often so dreadfully slow repainting, if you know what you're looking for, you can SEE that many a program does NOT actually use this facility for their "small windows", anyway)...

Messages in either case aren't particularly big...other than the obvious "Window ID" and "message type" data that `_has_` to be present, Windows just passes two 32-bit "generic" parameters, "co-ordinates" and a "time stamp" (the window procedure only sees the two "generic" parameters but the MSG structure has a few more things inside it)...X – because it has "bandwidth" to think about – just sends as many as is needed ("variable length"...if the "window ID" and "event type" is all that's needed, that's all it sends...if it needs some "parameters" then those are thrown in after it...the "event" structure isn't variable length, it's a "union" of all the possible "event" structures...but this isn't actually what's sent over the connection, just the data structure it plops the data into for the program to read :)...right, so it's not any "message queue" that's taking up the RAM...

If you're just creating one simple window and nothing fancy, then it seems doubtful that this could (at least, not justifiably so) be Windows' or X's own "internal" information (the "window class" stuff, internal variables for "management" of windows and so forth)...

So, where on Earth – in `_EITHER_` case – is all this RAM disappearing to?

This is `_poor_` implementation...neither Windows nor X justifiably need this much for solely doing what they need to do...it's "bloatware" syndrome responsible here...

And, with X, you can `_SEE_` this because, of course, the "standard" and "protocol" is open and available to all...there's nothing in the "X protocol" that seems to justify that any X server must be 500GB of disk space and run like a dog...indeed, when I read "network transparent", I thought "oh-oh...there's going to be a bunch of overly complicated nonsense at the bottom!"...but I took a look and, nope, it's refreshingly simple...

So simple, in fact, that it really can be described in a small space...right, the method of sending the "requests" and "events" (collectively, I'll call them "messages", like Windows' terminology :) is not covered...the "protocol" only covers the actual data sent...

When a "connection" between a program and X is opened for the first time, a small "packet" is sent that specifies the "endianness" and version numbers...the "endianness" is used for all "messages" sent thereafter over this connection..."version numbers" do the usual things of letting the program / server know that it's vastly "out-of-date" compared to what it's working with...servers can "scale" themselves according to the "version number" they receive to behave like a previous version...and so on and so forth...

Right, then any "request" you send the server is an 8-bit "major opcode" byte (specifying the "ID" of what you're asking the server to do with this "request")...a 16-bit "length" field follows to specify how many additional bytes follow for the "parameters"...the parameters follow, `_IF_` there are any involved with that type of message...the "major opcode / length" header

thing is padded to 4 bytes...the "parameter" stuff after it is also padded to 4 bytes...an "alignment" thing to make it work in neat 32-bit "chunks"...

So, for example, with "DestroyWindow", you send a byte with the "DestroyWindow" major "opcode" (just a "message ID", basically :)...then a "length"...this message only has one parameter, though...the "window ID"...so, send those along...and, well, that's basically it...

As I say, it's refreshingly simple...the "network transparency" is coming from the fact that all "servers" – whatever machine they are on – all use the exact same data format...and that all the data is naturally stuff like "IDs", "opcodes", "co-ordinates" and such, which need no "interpretation" to be used on other machines (there's nothing "machine-specific" about this kind of data)...

XFree86 even ignores the possibility granted by the X protocol specification that it doesn't involve itself in `_HOW_` you want to send the data...it just uses "sockets" regardless (and a "sys_socket" syscall is in the Linux kernel itself)...you can use any kind of "IPC" for this, though...and one thing that the XFree86 people haven't realised is that X is not specifying what to use `_deliberately_`...for instance, if both "client" and "server" are on the same machine, then you can switch to using the more direct "IPC" (shared memory, message queues, etc. :) without the "sockets" overhead...it keeps out of this because this allows the `_best_` "transport" to be used, as applicable to the two machines involved...

Similarly, all communications in X are asynchronous and `_cache_`...the exact caching policy is, again, not specifically addressed...considered "implementation detail"...again, a deliberate choice because this means that, say, on a "local" connection, you can choose to go for "high responsiveness" (so that the user gets a speedy response..."bandwidth" is not really an issue when we're talking about processes on the same machine sending a few bytes back and forth to each other...so, make the "cache" small or even non-existent so that things happen straight away) but with a slow, remote connection, you might prefer "high throughput" instead (because all the messages are being sent over, say, a 56K modem connection, you don't want it to "flood" the connection with lots of pointless messages – consolidate a bunch of paint messages into just the one – and you "cache" them up to use the connection more optimally)...

I suspect, though, that if XFree86 just uses "sockets" regardless of the location of the machine, it probably also only has `_one_` caching policy too...setting it to some "generic" in-between of neither "high responsiveness" or "high throughput"...simultaneously pissing off both requirements at the same time...your X implementation is a bit slow? Probably because it has a crap "generic" caching policy and literally sits around for a second before sending anything, just to see if any other messages will turn up...which is actually useful over a "limited bandwidth" connection – a small delay in the "responsiveness" but makes use of the "connection" better – but is pointlessly unnecessary when both processes

alt.lang.asm: Re: Anybody here endure C/Cpp? (.h to .inc conversion)

are on the same machine (where even a "no cache at all" policy isn't necessarily a bad idea...it's what other non-network-transparent GUIs can only do, usually :))...

You see, lots of the blame is thrown onto X when it should be thrown onto poor implementations by "bloatware" programmers...

Indeed, a UNIX problem is its "C connection" because there's a lot of the "use libc blindly!" thing and "500MB of RAM? What's that? My development machine has 56GB of RAM!! Doesn't everyone have that much on their machines too? They don't? Ummm...ummm...they should...ummm...buy more RAM so that they do! What? Me actually change my program to work on an 'average' machine to suit my paying customers, my ultimate employers? No, sorry...that would involve being a 'professional' about things...I don't do that...it involves doing work...and 'developer time' is so precious – so the accountants tell me – that people should amputate their limbs and sell their children into slavery before I waste a second of MY 'developer time' on software quality"...

Worse, *NIX often also draws the "extremeists": "Users? That disgusting vile lot? If they don't know how to use all the voodoo 'vi' commands since birth (while editing their X configuration files using a soldering iron ;), then they are clearly ignorant animals who should be culled for the stain that their idiocy places upon the world! Hand me my shotgun...I'll sort out these great unwashed 'users'...how dare they be 'average'! How dare they not spend 6 billion dollars on hardware! How dare they not devote two decades of their life to being an anti-social techno-geek like I am!! KILL! KILL USERS!!" ...*bang*...*BANG*...hmmm, NOT a good idea to talk about "human-machine interfacing" or "good user interface design" with you, then, eh? ;))...

It's laughable, really...the state that the software industry has let itself get into...the problem is that some people think they somehow "solved" the "software crisis"...no, they papered over the cracks with crap HLLs and have never since peeked underneath the 97 layers of wallpaper to see what state the crack has become...indeed, the crack in the wall has actually become a gaping hole...they don't realise this, though, because with 97 layers of wallpaper on top of it, the wallpaper paste is keeping it solid...you could take the wall away and there's so many "layers" of wallpaper, they wouldn't realise it's gone on the other side at all...in fact, that's not quite as "metaphorical" as it sounds...plenty out there still think that assembly language programming "must" involve direct hardware programming, which makes it infinitely more complex than we use "libc" blindly everywhere...the fact that OSes and device drivers have completely changed this that they often don't even allow "direct access" at all, even if you want to do it...well, those "layers" of wallpaper over the crack...they would have no idea if you took the wall away completely...the "wallpaper" is now 3 feet thick, after all, so it's completely self-supporting..."virtual machines" running "scripts" running "VHLLs" running "scripts"...all so "distant" and slow that you could swap the machine for a little man hiding inside a PC box who's doing it all by hand

alt.lang.asm: Re: Anybody here endure C/Cpp? (.h to .inc conversion)

(drawing pictures with crayons and holding it up to a piece of glass, pretending to be a "monitor"...indeed, I often wonder if this is the actual "technology" Microsoft use for GDI ;)...they probably wouldn't notice the difference...

There IS a problem with X, though...namely, it was designed in 1985...and, in places, this shows quite clearly...

Also, it didn't deal with everything...that was part of the design..."modular"...that you can add things on by "extension"...but, for example, nothing in X itself about _sound_...hence, someone creates an "extension" for that...problem: 7 completely different people all decide to create "extensions" for it...oh dear...which one should be "standard"? It'll just have to be battled out between them...same situation with "toolkits", for sure...do you use Xt, Motif, GTK+, etc.?

The "flip side" of having a "modular" design and where lots of different parties are making "extensions"...you end up "flooded" by a hundred different things that are all, in fact, really doing the same job, albeit in slightly different ways...

Indeed, as NoDot mentioned before, I had thought about an idea of some kind of "X++"...well, you know, C went to C++...X could go to X++...same underlying design, possible "X compatibility" but addressing the "complaints" that X has...which, by the way, as I've been pointing out, aren't really complaints about X...they all are "peripheral" – poor implementations, confusions over having 27 "toolkits" to choose from, some "extension" doesn't work, crap "window managers", etc. – BUT it's X that gets the blame, as the "overall name" for what is, after all, lots of different things thrown together...but the simple concept would be redo X and, this time, there is sound support...there is a "standard" window manager...there is a "standard" toolkit...and it's implemented bloody properly for once...you know what I mean, yeah?

Programming X is actually easier and more logical than Windows...note, though, that X doesn't have native sound support, as an example...this is an "extension" onto "X"...and when I say "programming X", I mean programming the _native_ parts...the parts with the "easy and logical" design...those other bits aren't really "X" but are "extensions" by other people...regards whether those are easy and logical...yes, THAT is a different matter entirely...I say nothing about those in the statement about X...

And the separation of "GUI" from "window manager" from "toolkit" is a good idea...being "modular"...the problem there is that everyone and their donkey decides they can do a better "toolkit" and we end up with 27546 of the things in existence...confuses people rigid...

The fact that graphics are pixel-based (and arguably NOT "device-independent" because of this) is, of course, an equal complaint that could be made of Windows...this, in itself, is not a problem (when you

want to do various graphical things, you sometimes `_WANT_` your pixels to actually be literal pixels, not "virtual pixels" and all "vector-based"...so, really, that support `_should_` be there and it's logical enough that it's the `_lowest level_` interface...indeed, one should go further and have something more "DirectX"-like down there, I'd say), `_IF_` there'd been the "graphics library" to handle things when we've got a programmer who's writing a database or word processor or whatever and `_ISN'T_` interested in "clever graphics"...is NOT a "graphics programmer" at all...and so on and so forth...

In fact, blaming X is a little unkind, in the sense that X is `_ALSO_` the `VICTIM_` in all of this...it starts off well...this here, that there...okay, other people will come along and it'll grow from strength to strength...I've put in an "extensions" thing to support that, which does allow other new things to be added that fit in neatly with it all...great...what happens? The "peripheral" stuff becomes a "flood" of "toolkits" and "window managers" and "extensions"...everyone's scrambling all over each other...it's a panic! It's a mad rush! Users are trampled under foot...oh, the humanity! Oh dear, are those "bloatware" programmers we can see on the horizon? No, keep away! Keep away! What on Earth is this "window manager"? What do you mean it's "intuitive" that users have to dance the hockey-cockey and type in some "voodoo" sequence of characters just to close a window? Are you insane? Aaarrgh!! Can someone put "bouncers" on the door or something to keep all this "riff-raff" out?! Who the hell are you? A what extension? Oh, never mind...the carpet's been ruined with all these uncivilised "toolkits" spilling their drinks everywhere, anyway...who cares? Yeah, go on...just pour that bottle of red wine all over the carpet...it's such a mess in here, who's going to notice? Blah-blah-blah...

Plus, regards Linux, there are other things to note...there were no graphics drivers for Linux originally...worked just in text mode...so, the X implementation – not having any useful native OS graphics support – just opens up the I/O ports and start programming the hardware directly...ah, Linux has "SVGAlib" and "DirectFB" and other things added (again, another problem with a "mess of possibilities" to choose from ;)...ummm, too late...X does its own thing already...can we get these things to work together? Ummm, not really...

In a sense, Linus made a similar-ish kind of mistake that X originally did...it's like "oh, I won't add in the graphics / sound support at the moment...ummm, it can be added on later with revisions and 'extensions'"...so, X starts working without graphics drivers because Linus did have any of those to begin with...now, when graphics drivers are added in, we've got ourselves a little "mess" to sort out...logically, X should be using the Linux graphics drivers...instead, Linux has graphics drivers...not used for much but games...X has its own graphics...not much use for those games which use something completely different...dreaded "incompatibility"...and it's not because any of these things weren't designed properly...it's just a matter of really "BAD TIMING"...

alt.lang.asm: Re: Anybody here endure C/Cpp? (.h to .inc conversion)

And so on and so forth...but if by X we mean the `_standard_` then, yeah, I defend the statement: X is not perfect BUT in comparison to Windows' (design), it's an "angel"...

If this can't be easily seen running X then that's because Windows has one little advantage: "dictator" Gates...there is only one GDI, one shell, one DOS box, one DirectX, one "shell extensions", one "window manager", one "toolkit", etc., etc....oh, all these things are crap, mind you...Microsoft are masters of writing crap...their "design meeting" consists of a "boss", unshaven in a dirty vest with a cigarette hanging out of his mouth, shouting "don't bother me!! We don't discuss 'design' around here!!! Just make it up as you go along or something!! Can't you see? I'm trying to drink my bottle of vodka here and I don't want to be disturbed with this 'design' crap...design doesn't pay for my daughter's pony that she wants for Christmas and nags, nags about aaaall the bloody time...screw yer 'design', you big arty poof...one more word about 'design' and I'm firing you on the spot, ya good-fer-nuffin' hippy...just do yer bloody code – whatever the hell it is – and SHUT IT!"...*whip crack* ;)...

On the other hand, X was nicely done but, to be honest, unfinished...never mind, others will come along and it'll be "extended" in a nice, orderly and useful fashion, right? Nope, you've got to be kidding...Noah, get building your ark, here comes the flood – of "toolkits"...so, do you want Gnome or do you want KDE? UNICODE, what's that? Oh, ummm, we'll "hack" something out to try to cobble together a house of cards to support it or something...some time in the next century...maybe...

Truth is, they all suffer this...but it's more easily controlled and "hidden" with Windows because it's a "dictatorship" kind of deal...after all, there are all those "MZ stubs" in all those "PE" files because, hey, someone somewhere might be an idiot and try to run a Windows program under DOS 2.0...yeah, yeah...I know that crap's not needed anymore but we got "backwards compatibility", you understand? If one of our coders pisses all over the source code, we have to `_retain_` that for the next seven decades, "just in case" some program out there somewhere depending on pissed-up code...maybe...

With Linux and X, it's all "open source"...so everyone gets to see it...no brushing it under the carpet...Microsoft have far more crap, in fact...when you buy Windows, you're getting "pure DOS", V86 DOS "fake" in a box, Win16, Win32, DirectX (all nine versions), OpenGL, GDI, GDI+, WinG, COM, COM+, OLE (versions 1, 2, 3, etc., etc. ;), IE (all six versions), old text edits, new rich edits, even newer rich edits, even newer newer rich edits with IME 1 and IME 2 and IME 7 million, standard controls, common controls (how many versions? Probably 100 or something silly like that)...and we ain't even getting started yet...Windows is `_WORSE_` (in `_EVERY_` regard)...it's just more easier "hidden" when Microsoft have a "monopoly" and they "insist" that everyone "upgrades" to the newer crap...and people forget – though it's `_STILL THERE_` – that all these older versions and libraries no-one uses anymore still exist...but they do...I'm probably one of the last people who still regularly uses a `_Win16_` program all the time (an old

IDE...it does everything I want...why upgrade?)...Windows launches some "VDM" which launches some "WOW" (Windows on Windows) crap...

Windows is worse in every regard...but Windows "gets away with it" because of "monopoly"...even with all the "toolkits" and things thrown in, X doesn't have so many extraneous "versions" of 10,000 DLL files that no-one ever uses for anything...Windows is WORSE but Windows "gets away with it" because Bill decrees "everyone must upgrade!"...and the daft idiots actually do, every single time...

If programming X is like trying to find the square root of pi in roman numerals, then Windows must be like some binary complex number differential calculus in a twenty-seven dimensional problem to solve the meaning of existence in comparison...

Actually, hmmm, put like that...perhaps, in a "relative way", I agree...BOTH are extraordinarily crap GUIs...where did Paul Hsieh disappear to? He's written his own VESA-based GUI (with "transparent windows" and "shadows" too...yeah, released before Microsoft added their versions of these things to Windows...I did anti-aliased text before them myself...bitmap-based, though, not TrueType...but it was only a simple thing...bitmap fonts throughout...I would have made the vector-y stuff anti-aliased too, IF I'd gone so far as adding that...indeed, I should have "patented" it, eh? :)...he made a download for people to look at...it runs in high resolutions like 1024 x 768 and so forth with 256 or 16-bit or 24-bit colour depths...transparent windows (or parts of windows)...shadows...he did this ages back now...what's the "minimum specs"? Check it from his "readme":

- Runs on DOS, Windows
- Requires 486 compatible processor.
- Requires SVGA
- Requires Mouse.
- Requires about 4MB of memory.

Now, THAT is a more than reasonable set of "minimum requirements", eh? EXE file is only 385KB and this does include two example bitmaps to show how it works (a large Jodie Foster backdrop and some picture of a galaxy...neither of which are small so, chances are, the majority of that 385KB is for these JPEGs, as they are actually inside the program itself)...

There's no reason at all, as I say, why GUIs are this big, this complicated, this slow, this "bloated" and this crap...it's just poor design / implementation / programming from the best "bloatware" programmers in the business...

Beth :)