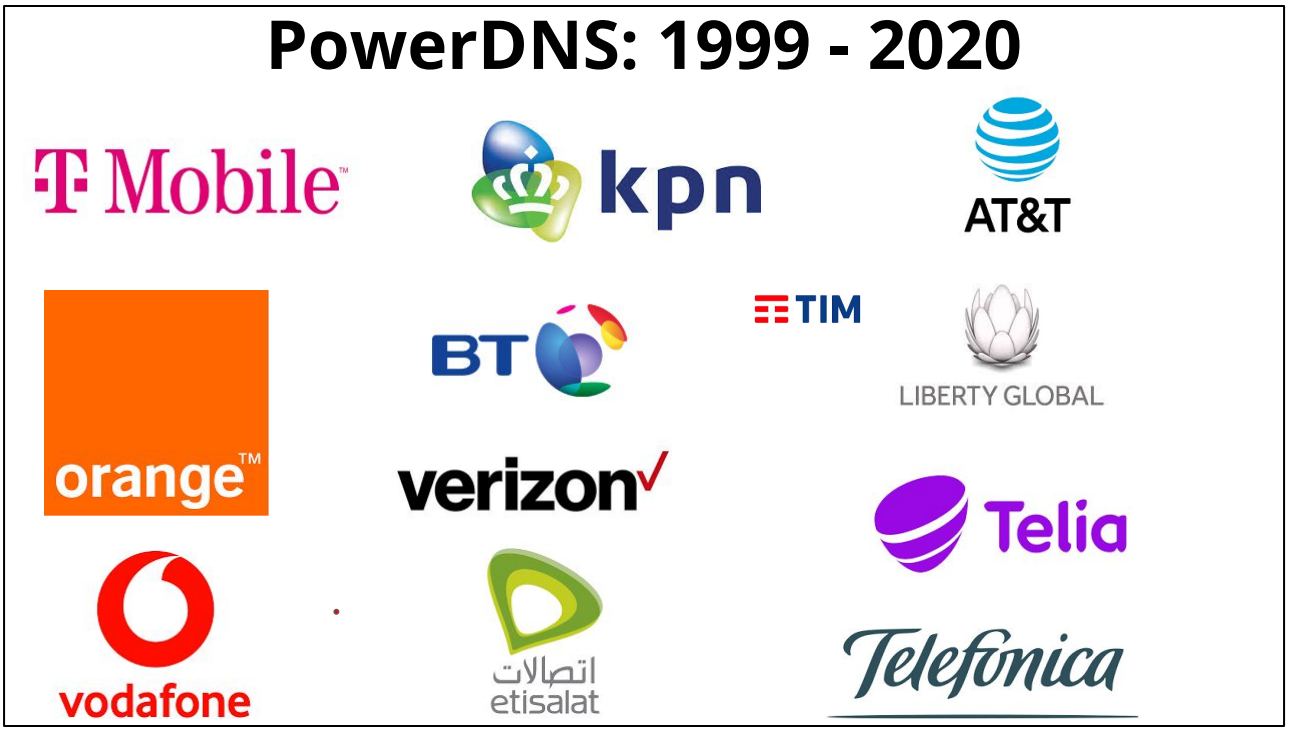
Software

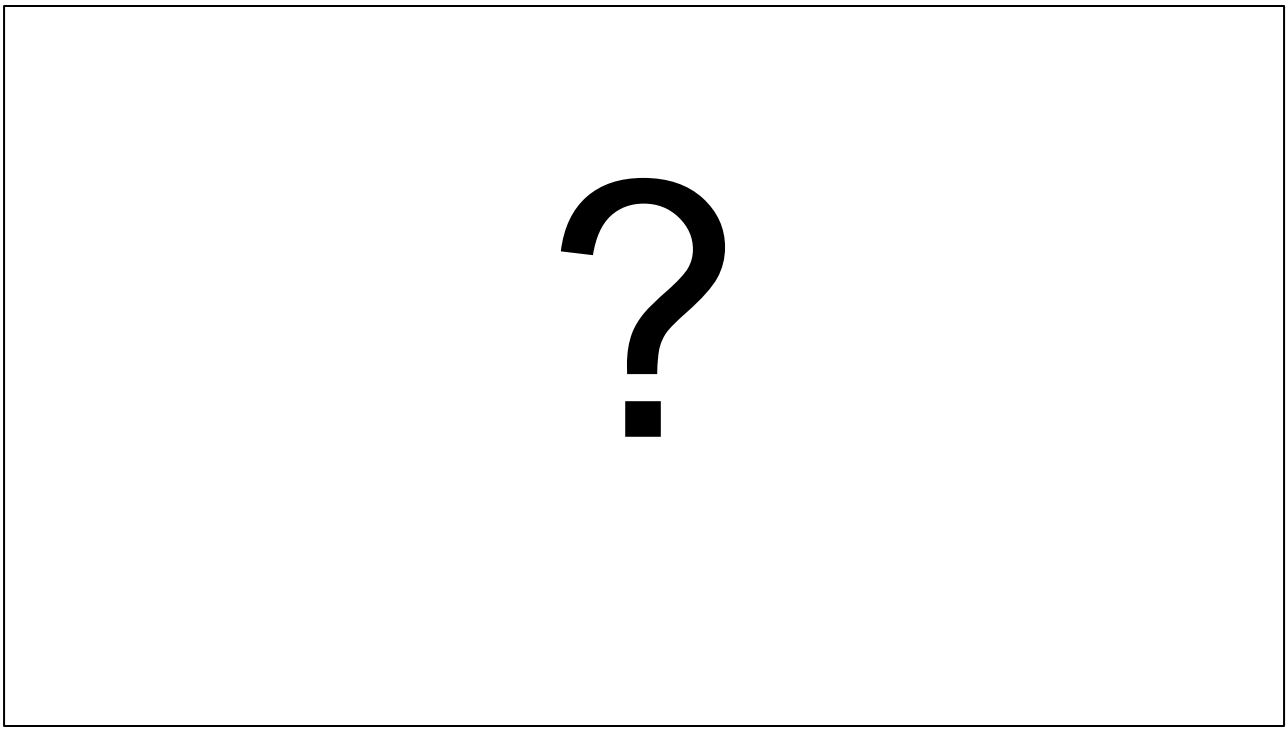
bert@hubertnet.nl / https://berthub.eu/
https://berthub.eu/les/



Places my company worked with or was involved with negotiations & contracts

The internet gave me an award, I'm an officially recognized nerd.
https://berthub.eu/articles/posts/nluug-award/

I was a member of a panel with two judges that had to rule on all warrants from the dutch intelligence/security agencies



I wasn’t sure exactly what to talk about. The request was ‘software, source code, and lots of examples’ so here goes

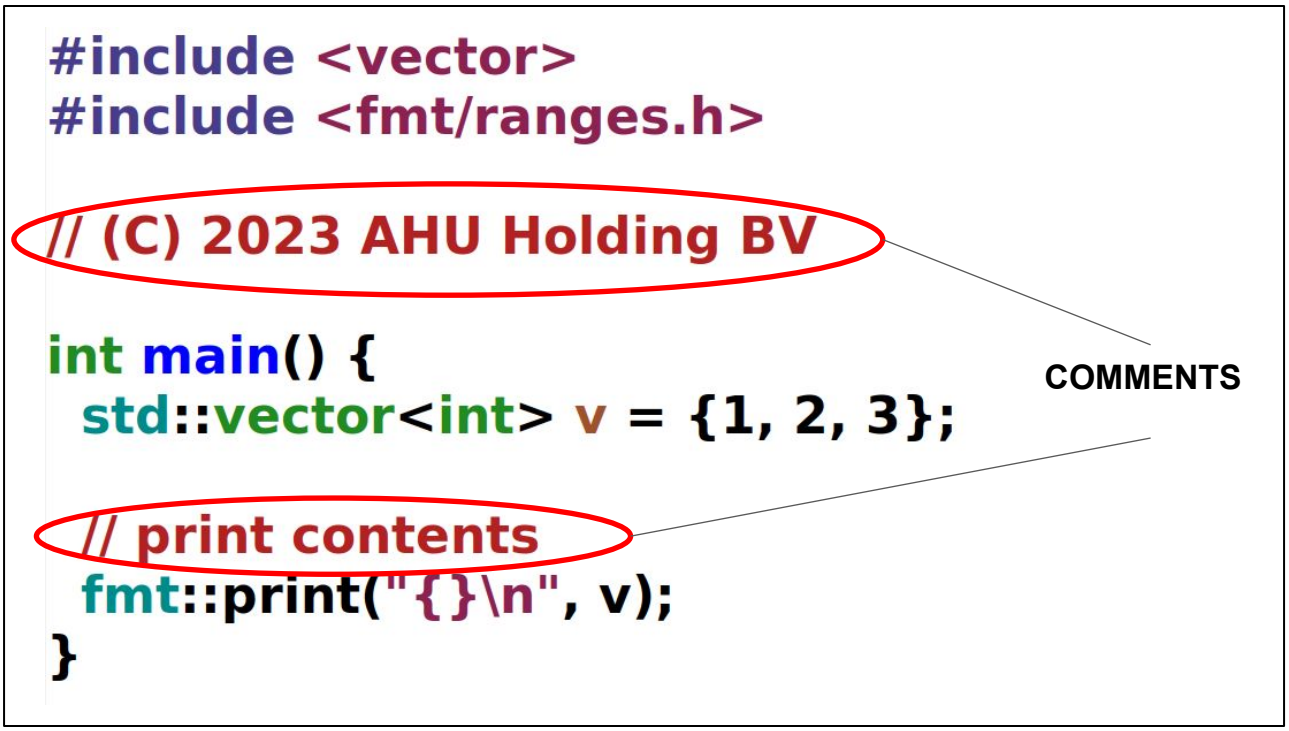
```cpp
#include <vector>
#include <fmt/ranges.h>

// (C) 2023 AHU Holding BV

int main() {
  std::vector<int> v = {1, 2, 3};

  // print contents
  fmt::print("{}\n", v);
}
```

This is a representative bit of code, showing 5 common elements

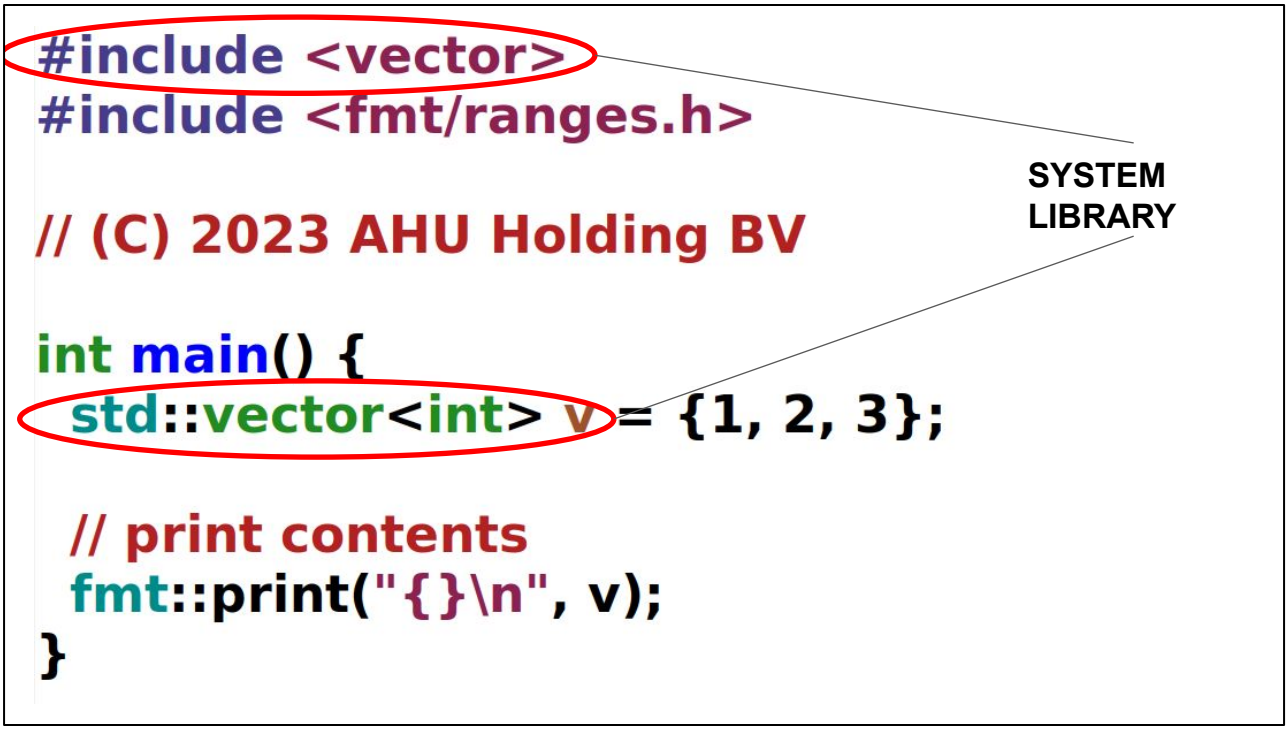
This is the output of the computer program - it prints the 1,2,3 that was stored in variable v

HI	LO-NIBBLE															
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	BRK impl	ORA X,ind				ORA zpg	ASL zpg		PHP impl	ORA #	ASL A			ORA abs	ASL abs	
1-	BPL rel	ORA ind,Y				ORA zpg,X	ASL zpg,X		CLC impl	ORA abs,Y				ORA abs,X	ASL abs,X	
2-	JSR abs	AND X,ind			BIT zpg	AND zpg	ROL zpg		PLP impl	AND #	ROL A		BIT abs	AND abs	ROL abs	
3-	BMI rel	AND ind,Y				AND zpg,X	ROL zpg,X		SEC impl	AND abs,Y				AND abs,X	ROL abs,X	
4-	RTI impl	EOR X,ind				EOR zpg	LSR zpg		PHA impl	EOR #	LSR A		JMP abs	EOR abs	LSR abs	
5-	BVC rel	EOR ind,Y				EOR zpg,X	LSR zpg,X		CLI impl	EOR abs,Y				EOR abs,X	LSR abs,X	
6-	RTS impl	ADC X,ind				ADC zpg	ROR zpg		PLA impl	ADC #	ROR A		JMP ind	ADC abs	ROR abs	
7-	BVS rel	ADC ind,Y				ADC zpg,X	ROR zpg,X		SEI impl	ADC abs,Y				ADC abs,X	ROR abs,X	
8-		STA X,ind			STY zpg	STA zpg	STX zpg		DEY impl		TXA impl		STY abs	STA abs	STX abs	
9-	BCC rel	STA ind,Y			STY zpg,X	STA zpg,X	STX zpg,Y		TYA impl	STA abs,Y	TXS impl			STA abs,X		
A-	LDY #	LDA X,ind	LDX #		LDY zpg	LDA zpg	LDX zpg		TAY impl	LDA #	TAX impl		LDY abs	LDA abs	LDX abs	
B-	BCS rel	LDA ind,Y			LDY zpg,X	LDA zpg,X	LDX zpg,Y		CLV impl	LDA abs,Y	TSX impl		LDY abs,X	LDA abs,X	LDX abs,Y	
C-	CPY #	CMP X,ind			CPY zpg	CMP zpg	DEC zpg		INY impl	CMP #	DEX impl		CPY abs	CMP abs	DEC abs	
D-	BNE rel	CMP ind,Y				CMP zpg,X	DEC zpg,X		CLD impl	CMP abs,Y				CMP abs,X	DEC abs,X	
E-	CPX #	SBC X,ind			CPX zpg	SBC zpg	INC zpg		INX impl	SBC #	NOP impl		CPX abs	SBC abs	INC abs	
F-	BEQ rel	SBC ind,Y				SBC zpg,X	INC zpg,X		SED impl	SBC abs,Y				SBC abs,X	INC abs,X	

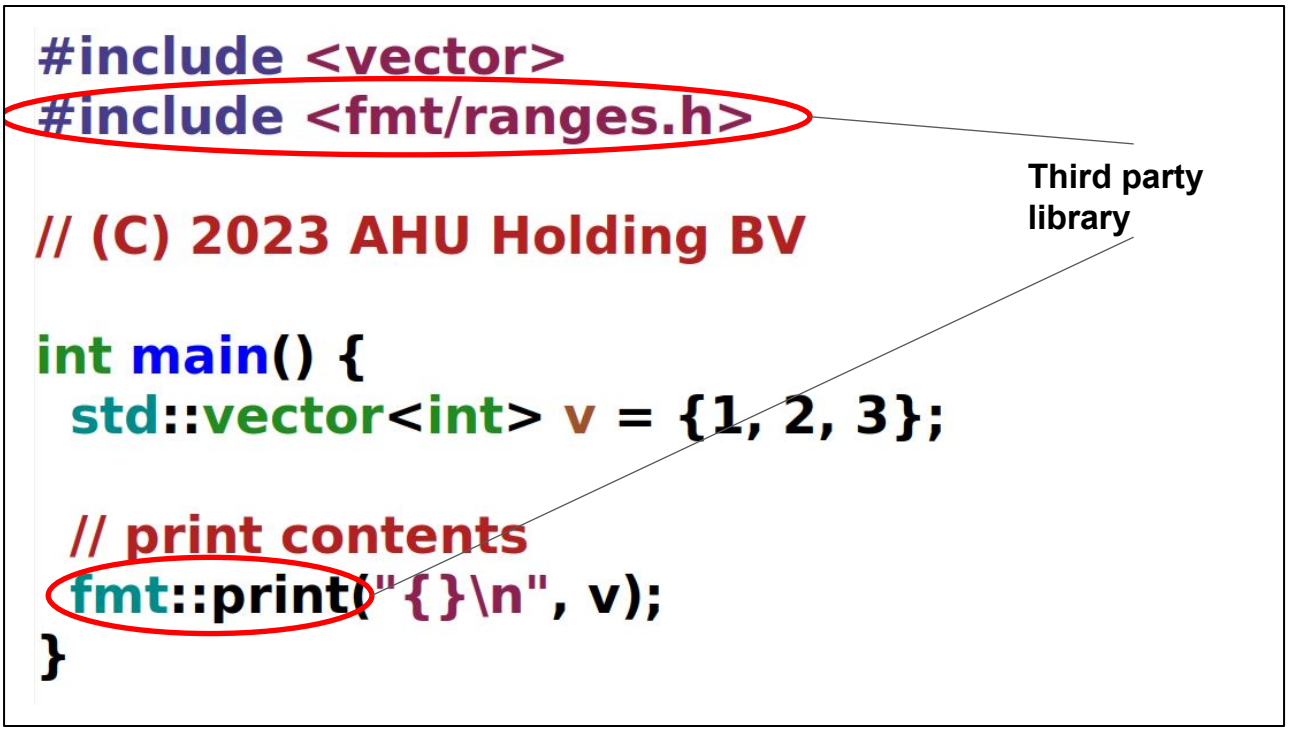
Computers don’t actually run source code directly (mostly). They run machine code, as shown in this table. Machine code is easy to read for computers and hard to write for people. Source code, as shown earlier, is hard to read and write for both computers and people. So it is a compromise where everyone loses.



Comments! They are full of problems. For one, the computer does not read them. So the comments can be full of wise words, but the code can then do different things. Of special note is that programmers for a variety of reasons put legal statements in comments. These include incorrect copyright statements, but also license texts that may or may not apply. If you audit the source code of any company you'll find lots of code with © other companies in there, but that is not itself a problem. That code may be licensed freely. Comments however are often wrong or misleading. They can help you figure out where code came from though, which could be helpful.

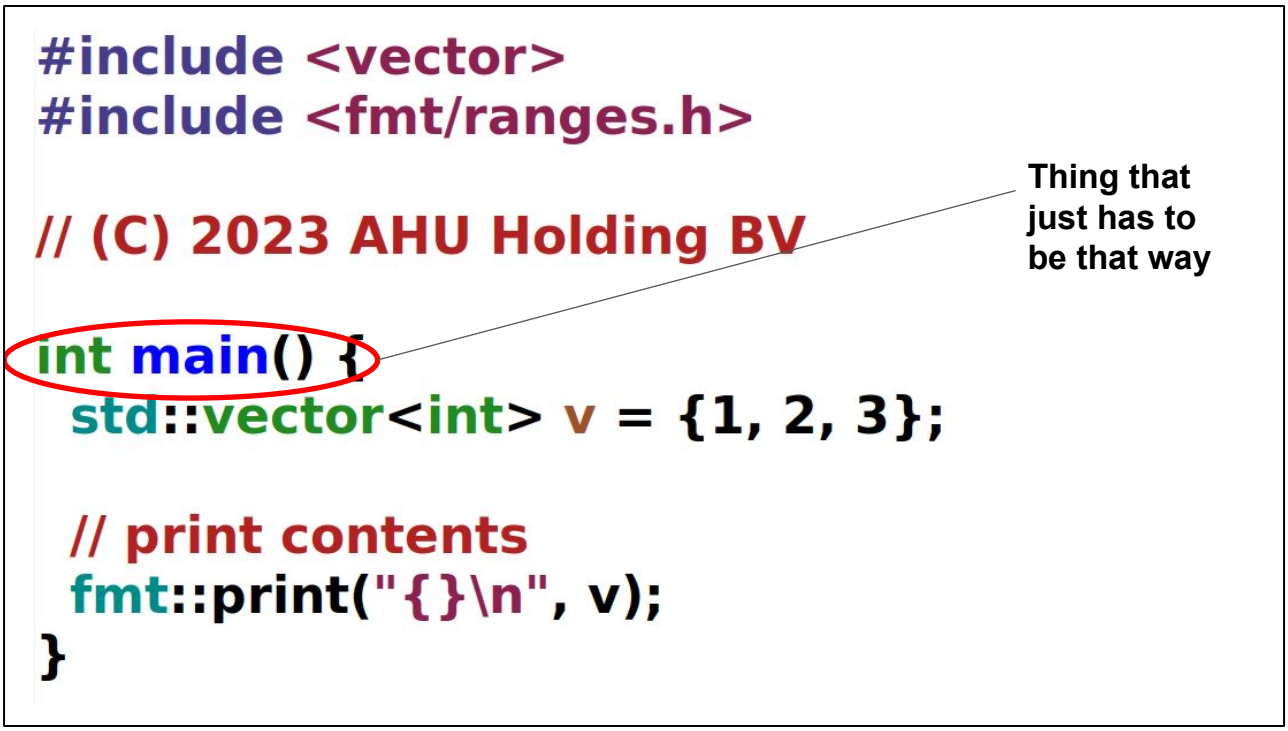


Source code will ALWAYS build on ‘system libraries’. So to store numbers, as is shown here, you’ll rely on the system code. This system code is always licensed such that when you incorporate it in your computer code, the resulting computer program (executable, binary) is *unencumbered*, so you get a license to ship programs containing system code. It is almost never a thing to worry about.



This tiny computer program relies on a third party product called ‘fmt’, mostly developed by a Meta employee. Fmt is licensed very freely so we can use this code without any worries. To spot ‘fmt’ as third party requires a good eye and domain knowledge - in this case the #include <fmt/> prefix and the fmt:: later on. Third party code becomes part of your product, and with that your computer program becomes a derived work. You should care very deeply about the provenance and licensing of third party code.

There is declared and undeclared third party code. Declared is as above when it is clearly something that is ‘included’ from somewhere else. Staff may however also have copy pasted third party code into your source code, and not been explicit about this. There are companies that will nevertheless recognize this third party code for you, which you may have to do in an M&A situation.



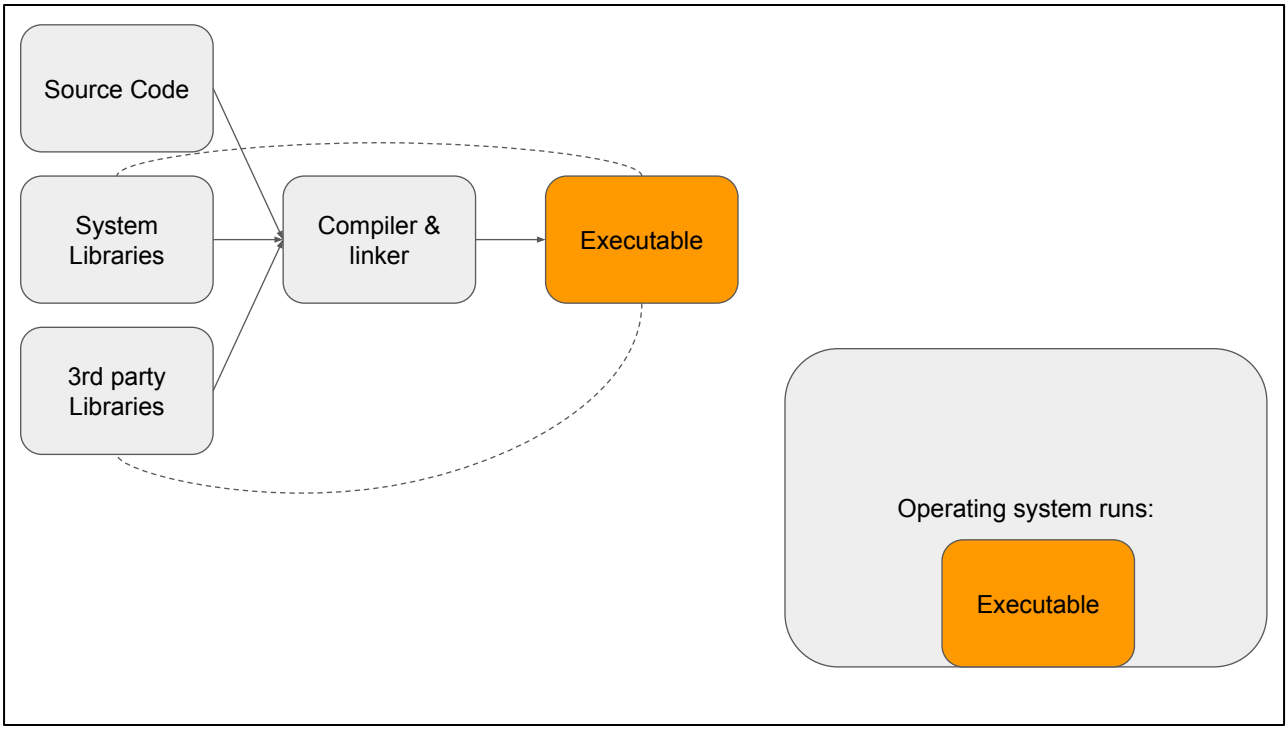
Some code has no “originality” or artistic freedom and just has to be like that. This means that 100% identical code can be found between Apple source code and Red Hat source code without this being a copyright violation. We’ll touch on this in a later slide.

Elements of source code

- **Comments**: no influence over what the code DOES (usually)
  - May include attempts at legal statements & disclaimers
  - Suitable for forensics: where did this code come from?
- References to **system libraries / components (“2nd party”)**
- References to **third party libraries**
- **Ritualistic invocations, technical** details (“main()”)
- **Actual code instructions** for this computer program

Access to source code means **nothing**. Is like having a copy of a book. No rights are conferred.

Source code is not as great as people think. Not the crown jewels.



Schematically - you need to add system libraries and 3rd party libraries to your source code before it is complete. That combination is then processed by a compiler & linker (or interpreter), and this delivers a program you can start, often called an ‘executable’. This executable meanwhile still also can’t run itself, it needs an operating system to do so.

Dynamic libraries: kernel interface, C++, compiler internals, C library, math library, dynamic system loader

```
ahu@brul2:~/git/code-tour$ ldd ./hello
        linux-vdso.so.1 (0x00007ffc6b981000)
        libstdc++.so.6 => /lib/x86_64-linux-gnu/libstdc++.so.6 (0x00007f9018a00000)
        libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f9018ca8000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f9018600000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f9018919000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f9018d0f000)
```


Static library: fmt formatting library

```
ahu@brul2:~/git/code-tour/ext$ ls fmt-10.1.1/src/fmt.cc fmt-10.1.1/src/os.cc
fmt-10.1.1/src/fmt.cc  fmt-10.1.1/src/os.cc
```

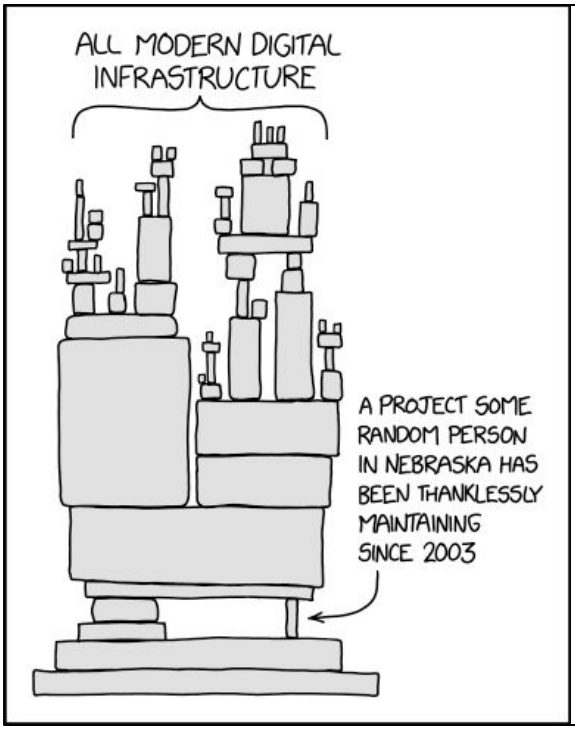

This slide was by special request: I was asked to comment on static and dynamic linking. However, if this is an issue for your company, you'll find that this is very complicated. I mostly show these numbers to indicate how difficult it all is.

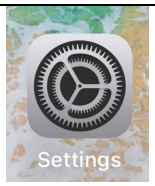
These libraries are all ***dependencies***. Each "dep" requires a ***license*** that allows you to ship an executable.

There are very permissive open source software licenses that allow you to do almost anything with the source code covered by it. But there are also many other more complicated licenses.

If you ever need to do battle over impact of static or dynamic linking or **plugins**: **hire the best**.
(or give up)

Some licenses have different rules for statically and dynamically linked dependencies. This stuff is exceptionally difficult. If you are ever forced to play this game, hire the very best technical support you can find. The cost of messing it up is huge.





On an iPhone, go to:

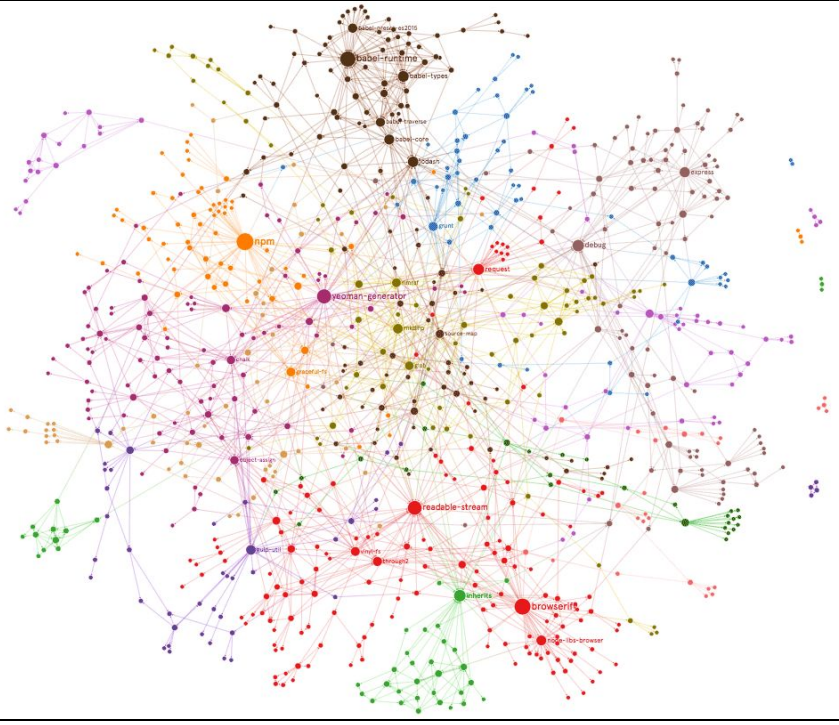
- Settings
- General
- Legal & Regulatory
- Legal Notices.

Try to scroll to the end!

https://xkcd.com/2347/ - modern software depends on soooo much other software. The picture on the left is accurate. I'm so happy many of you joined the exercise on the right to try to scroll down to the end of the iPhone 'legal notices' about third party software. You can't reach the end. That page is thousands of screens long! Android has a similar page somewhere.

1000s of dependencies!

Changes every day also!



Modern ‘node.js / npm’ based software easily racks up thousands of dependencies. These are gathered dynamically when the software is assembled. This list of dependencies changes daily. You need special tooling to create these graphs. The number of dependencies is so large that you can’t manually audit them, or even investigate by hand.

Things to watch out for

- No visibility in what is being shipped
- You are on the hook legally though - need explicit licenses
- Who knows if dependencies own the code they ship
- **Effectively you can’t keep track**
- Of specific note for M&A

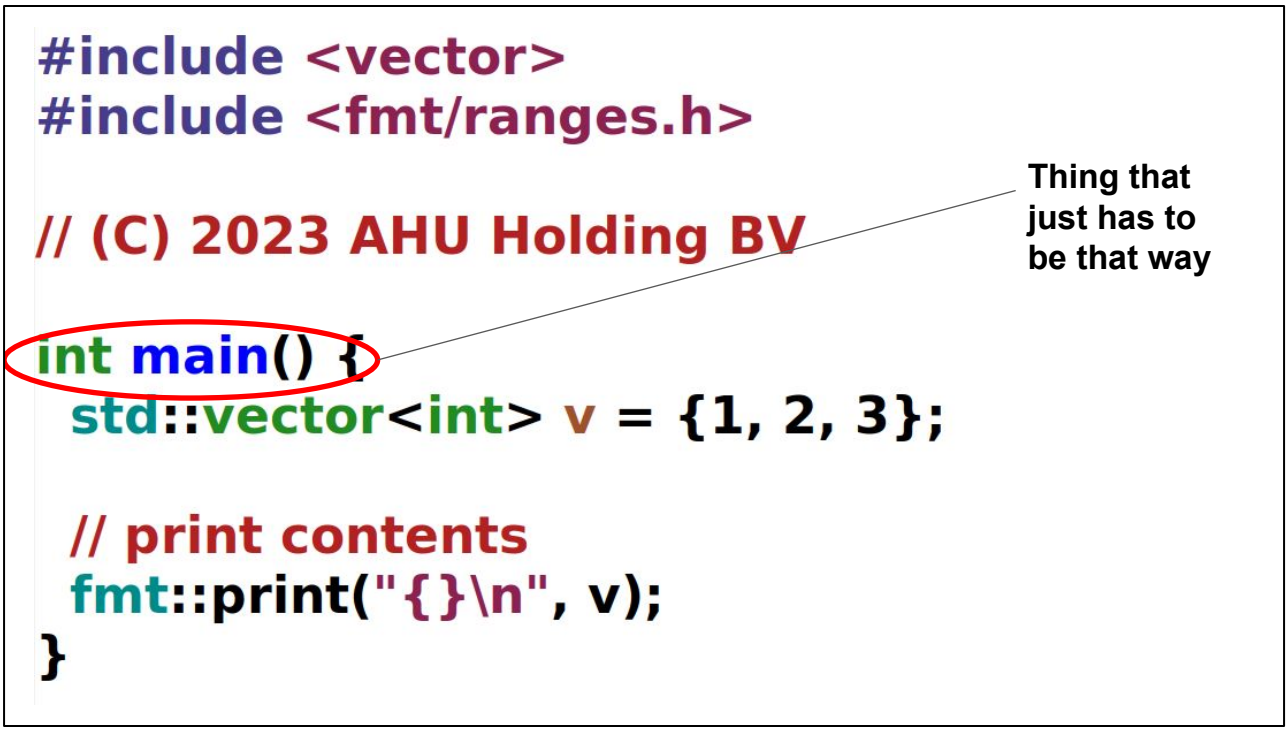
"The Skype **founders** apparently **retained** the service's peer-to-peer sharing technology when they sold to eBay for $2.6bn in 2005. (Which, of course, begs the question why eBay would pay all that money without ensuring **they own the entire platform)**."

https://www.theregister.com/2009/06/26/skype_trial_could_hurt_ipo/
https://web.archive.org/web/20121118205740/http://blogs.skype.com/en/2009/11/jolti d_settlement.html fascinating stuff. I've experienced something similar twice in my own personal contractual dealings.

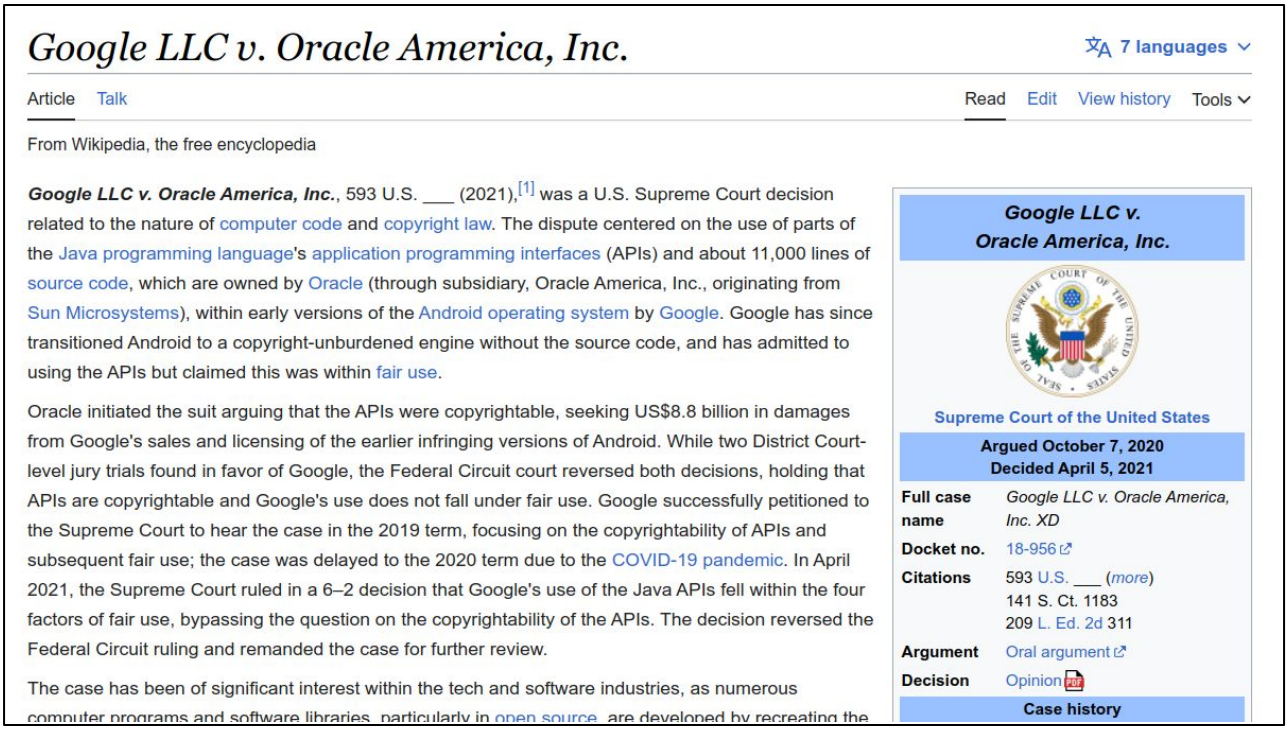
NPM ERR!

How one programmer broke the internet by deleting a tiny piece of code

https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code
https://www.theregister.com/2016/03/23/npm_left_pad_chaos/



So what can be copyrighted? I noted earlier that “some bits of code” just have to be like that. As programmers we’ve always been convinced this was true. For example, an internet data packet has a fixed *header* that tells you where a packet comes from and where it has to go. A technical description of that header had always been assumed to be a technical detail and not artistic expression. However…



Google LLC v. Oracle America, Inc.

7 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Google LLC v. Oracle America, Inc., 593 U.S. ___ (2021),[1] was a U.S. Supreme Court decision related to the nature of computer code and copyright law. The dispute centered on the use of parts of the Java programming language's application programming interfaces (APIs) and about 11,000 lines of source code, which are owned by Oracle (through subsidiary, Oracle America, Inc., originating from Sun Microsystems), within early versions of the Android operating system by Google. Google has since transitioned Android to a copyright-unburdened engine without the source code, and has admitted to using the APIs but claimed this was within fair use.

Oracle initiated the suit arguing that the APIs were copyrightable, seeking US$8.8 billion in damages from Google's sales and licensing of the earlier infringing versions of Android. While two District Court-level jury trials found in favor of Google, the Federal Circuit court reversed both decisions, holding that APIs are copyrightable and Google's use does not fall under fair use. Google successfully petitioned to the Supreme Court to hear the case in the 2019 term, focusing on the copyrightability of APIs and subsequent fair use; the case was delayed to the 2020 term due to the COVID-19 pandemic. In April 2021, the Supreme Court ruled in a 6–2 decision that Google's use of the Java APIs fell within the four factors of fair use, bypassing the question on the copyrightability of the APIs. The decision reversed the Federal Circuit ruling and remanded the case for further review.

The case has been of significant interest within the tech and software industries, as numerous computer programs and software libraries, particularly in open source, are developed by recreating the

Google LLC v. Oracle America, Inc.

Supreme Court of the United States

Argued October 7, 2020
Decided April 5, 2021

Full case name	*Google LLC v. Oracle America, Inc. XD*
Docket no.	18-956
Citations	593 U.S. ___ (*more*) 141 S. Ct. 1183 209 L. Ed. 2d 311
Argument	Oral argument
Decision	Opinion

Case history

Enter Oracle. Oracle thought that Google using such technical details was a copyright violation. After a decade of lawsuits, the US supreme court sidestepped really ruling on the issue and called it ‘fair use’. But a cloud still hangs over US copyright right now. The situation in Europe appears to be clear though.

Further things to watch out for

- Upcoming: EU Cyber Resilience Act, Product Liability Directive
- Put you up the hook for whatever you ship
  - Including dependencies!
- Software becomes a “real product”
  - CE

This is the future. You will be on the hook for whatever software you ship. And customers can hold you liable, just as if you were selling a toothbrush. This will become reality in 2026 but the time to start preparing is NOW.

Software

bert@hubertnet.nl / https://berthub.eu/
https://berthub.eu/les/