Introduction to free and open source software

Fundamentals of reproducible research and free software

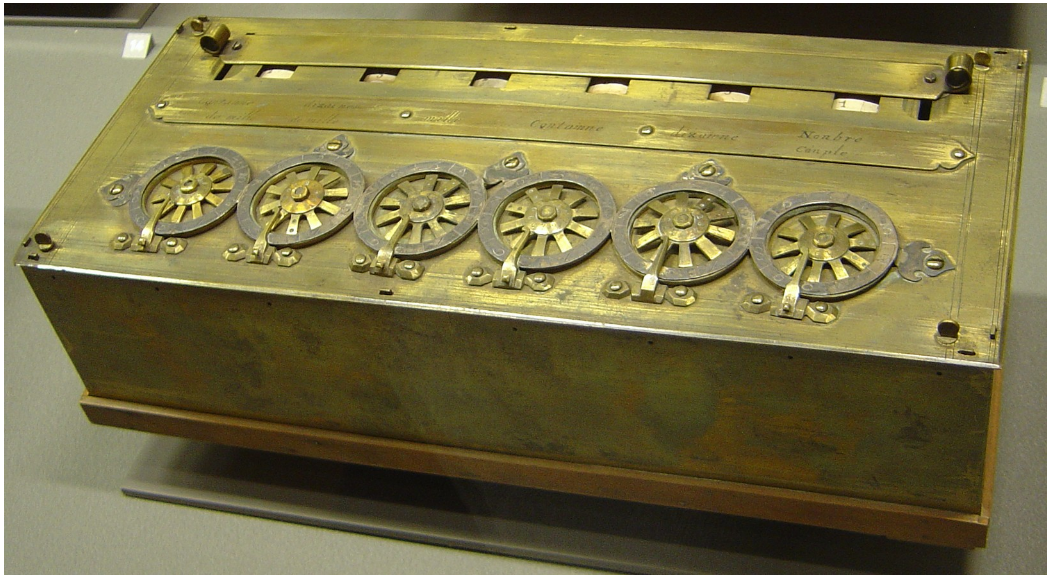
Miguel Colom
http://mcolom.info

Some milestones in computing

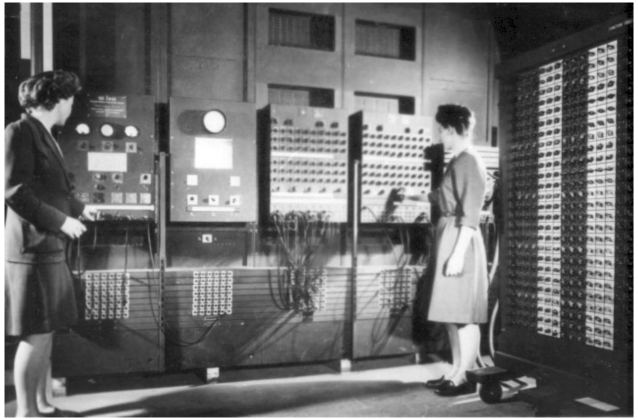
- **Ramon Llull** (1308) proposes a method for automatic reasoning. "Thinking machine".
- **Blaise Pascal** (1642): adding machine
- The **ENIAC** (**E**lectronic **N**umerical **I**ntegrator **A**nd **C**omputer) machine is built (1946)
- **Joseph Jacquard** (1801): textile loom with punch cards → first programmable machine?
- **Charles Babbage** (1833): analytical engine. Later Ada Lovelace proposed using punch cards to make it programmable
- **George Boole** (1854): *boolean* algebra
- **Frederick Guthrie** (1873), **Thomas Edison** (1880): discovery of the thermionic emissions, the base of vacuum tubes

Some milestones in computing

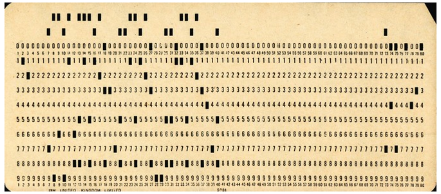
Llull’s *thinking machine*


Pascal *adding machine* (“Pascaline”)

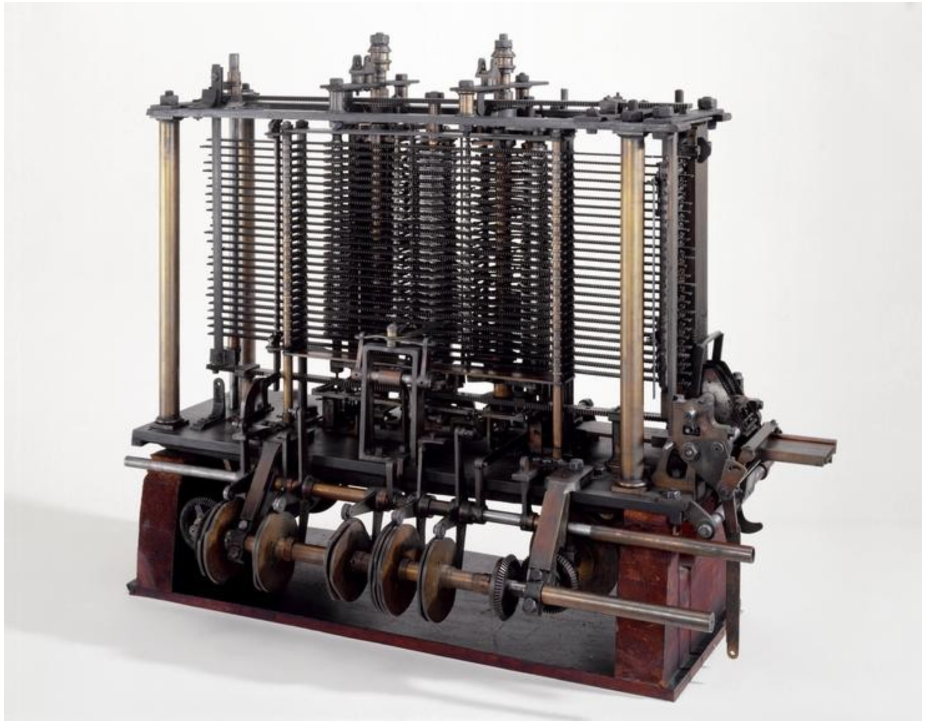
Some milestones in computing

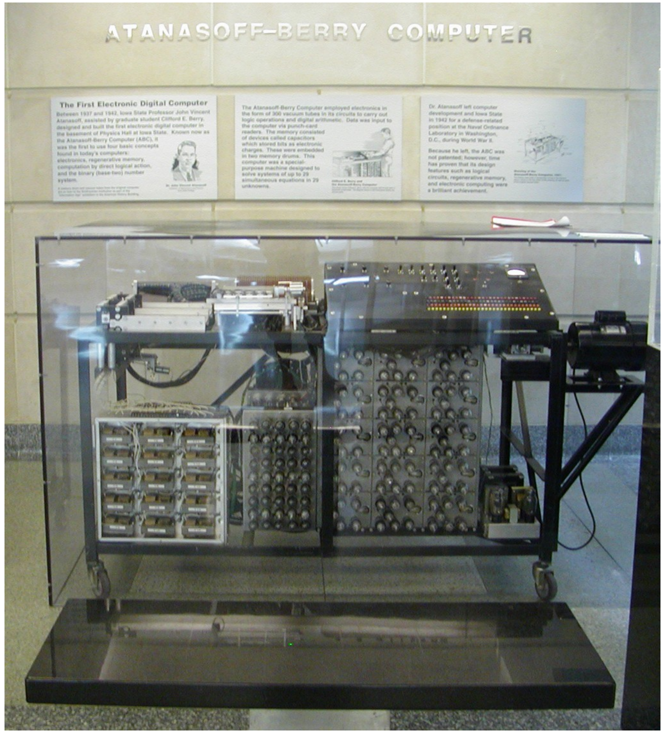

ENIAC computer

Jacquard textile punch card machine



Some milestones in computing





Babbage analytical engine

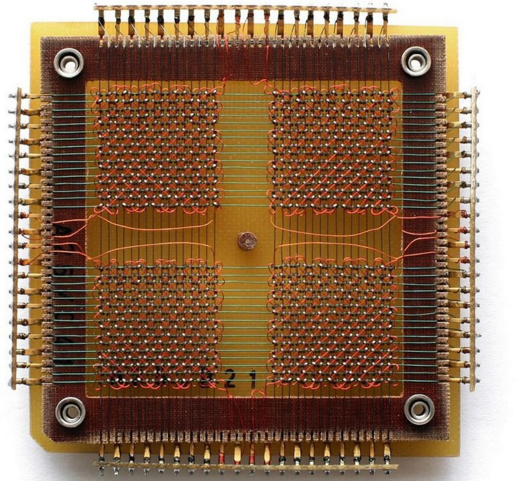
A computer based on vacuum tubes

Some milestones in computing

- **Alan Turing** (1936): basis of automatic programming. Turing machine. Stop problem. Turing test. And many theoretical developments.
- **John von Veumann** (1945) proposes the von Neumann architecture
- **John Bardeen** (1947): invents the BJT transistor. Start of microelectronics!
- **Kathleen Booth** (1948): invents the first assembly language, for the Automatic Relay Calculator (ARC)
- **An Wang** (1949): invents the core memory
- **Grace Hopper** (1952): idea of using high-level languages and compilers
- **Remington Rand**, US company (1953): releases the source code of the A-2 system for **UNIVAC**. First free/open source software?
- **ARPANET** (1969) implements TCP/IP. First pre-internet node at UCLA.

Some milestones in computing

A replica of the first BJT



Core memory

→ Curious video to learn about core memory:
https://www.youtube.com/watch?v=AwsInQLmjXc

Some milestones in computing

- First **cheap**, **low-power**, **small CPUs**: Intel 4004 (1971), 8008 (1972), Zilog Z80 (1978)
- **IBM** (1981) invents the "**personal computer**” (**PC**)
- **Richard Stallman starts the GNU project (1983), the basis of free software**
- **Yann LeCun** (1988) applies backpropagation to a deep neural networks: start of deep learning. Wei Zhang (1988) applies it to a CNN.
- The **WWW** starts at CERN, by **Tim Berners-Lee** (1990)
- **Linus Torvalds** starts **Linux** (1991)

Some milestones in computing

The first PC

Some milestones in computing

- **IBM** (2001) introduce the first **multi-core CPU**, Power4 architecture
- **IBM** (2021) proposes a **quantum computer** with 100 qubits
- **David Baker**, **Demis Hassabis**, and **John Jumper** (2024): **Nobel prize** to in **Chemistry** to reveal computationally the structure of proteins with the **AlphaFold2** system

Let's go back in time…

- **Richard Stallman starts the GNU project (1983), the basis of free software**

Image: Mikhail Leonov/Shutterstock

Free software and the GNU project

- **Richard M. Stallman**, activist and computer scientist at **MIT**
- Argues that one should have the right to share SW with their neighbors, to **study** it, **learn** from it, and makes **changes**.
- Not just for practical matters, but a **moral value**. <u>**Freedom**</u> of the user.

Free software and the GNU project

- In 1985 he publishes the ***GNU Manifesto***. Goal: write a "**free**" (as in freedom) version of **UNIX**, an existing proprietary closed-source OS.
- Right after, he starts the **Free Software Foundation** (FSF): legal infrastructure for the free-software movement. Namely: legal licenses, such as GPL.

- Nowadays: the GNU project is a ***mass collaborative initiative for the development of free software*****.**

The four freedoms of free software

- Defined by the FSF (1986)

- **Freedom 0**: to **use** the program *for any purpose*
- **Freedom 1**: to **study** how the program works, and **change** it to make it do what you wish
- **Freedom 2**: to **redistribute** and make copies so you can help your neighbor
- **Freedom 3**: to **improve** the program, and **release** your contributions to the public, so that the whole community benefits

→ **Any software which complies with this is free software** ❗

Free and open-source software

- Is it **the same**?
- Definition of open-source software by Red Hat: *Open source software is code that is designed to be publicly accessible—anyone can see, modify, and distribute the code as they see fit.*

open source initiative®

Free and open source software

- **Compatible** with the **four freedoms** of free software!

    → Therefore, **open source is free software**

Free and open source software

- Is it **the same**? **No**.

- What's the **difference**?
  - **Open-source** movement: focused on a software **development model** which encourages collaboration. Public availability → better software.
  - **Free-software** movement: the most important is the **freedom** of the user, rather than the product itself or any other technical considerations.

Other kind of software

- **Freeware**: distributed **without demanding a fee** for its usage. Free as in "*free beer*", not as in "*freedom*"**.** Gratis.
- **Shareware**: distributed without any cost but for a specific **evaluation period** only.
- **Abandonware**: **no active development**, and the author is **not taking care of it**. Also, if the **copyright** status is **not clear**.

→ These are **NOT free software!**

- The "freedoms" and these are just definitions or interpretations, without any legal value → Need of a **legal framework**.

Software licenses

- **Legal** *binding* between software and its usage
- It might include:
  - Can you redistribute the software? Any restrictions?
  - Can you modify the program in any way?
  - Can you use the program as a component in a larger system?
  - Can you make copies of the software?
  - Can you access the source code? Can you try to decompile it?
  - Can I use the software in any context? For example, medical or as part of a critical system (such a nuclear central).
  - Warranties for the final use. Hold liable.
  - And any other clauses.

Classification of software licenses

- **Public domain:** the author **abandons** its **ownership**. Therefore, no possibility of copyright, trademark or patenting. No restrictions, nor any freedoms: it can be modified, distributed, sold, sublicensed, made closed-source, etc. Examples: ELIZA (1966), BLAS (1979), ImageJ (1997). **NO FREE SOFTWARE**
- **Copyleft FOSS** license: any free or open source license. Copyright retained, right to copy, modify, re-distribute. **Can't be sub-licensed**. Example: GPL license.
- **Permissive FOSS** license**:** similar to copyleft, but it allows **sub-licensing**. Example: BSD licenses.
- **Proprietary** license: in general, any non-FOSS license. Normally they **forbid** making copies, modifying, redistributing, sub-licensing, … Example: Microsoft Windows End User License Agreements (EULA).

Example of public domain license: ELIZA (1966)

- **ELIZA:** a "psychotherapist" NLP computer program, by Joseph Weizenbaum.
- Complete original source code found and preserved in 2021.
- Licensed under a Creative Commons CC0 public domain license.
- https://creativecommons.org/publicdomain/zero/1.0/deed.en
- → https://creativecommons.org/publicdomain/zero/1.0/legalcode

The person who associated a work with this deed has dedicated the work to the public domain ***by waiving all of his or her rights*** *to the work worldwide under copyright law, including all related and neighboring rights, to the extent allowed by law.*

You can copy, modify, distribute and perform the work, even for commercial purposes, all without asking permission.

Example of a free-software license: GPL (1989)

- GNU **G**eneral **P**ublic **L**icense
- Free Software Foundation
- **Allows**: **commercial use**, modification, distribution, place warranty, use patent claims
- **Forbids**: **sublicensing**, hold liable
- **Must**: state changes, disclose source, include license, include copyright, include install instructions.

https://www.gnu.org/licenses/gpl-3.0.html

Example of a free-software license: AGPL (2007)

- Affero GPL. Free Software Foundation
- Intended for software running on networks (websites, cloud, …)
- **Similar to GPL3, adding that the source code must be distributed on the web**
- An example of website using A-GPL: IPOL

https://www.gnu.org/licenses/agpl-3.0.en.html

Example of an open-source license: LGPL (1991)

- Lesser GPL. Free Software Foundation
- **Not recommended** by FSF: https://www.gnu.org/licenses/why-not-lgpl.html
- **Allows programs using LGPL software not to distribute their source code**
- After all, a free-software license. Very special case. **To avoid**.
- Intended for libraries

https://www.gnu.org/licenses/lgpl-3.0.html

Example of an open-source license: MIT (1991)

- Short text
- Also known as *"Expat license"*.
- **Allows**: commercial use, modification, **sub-licensing**
- **Forbids**: hold liable
- **Must**: include license, include copyright

Example of an open-source license: MIT (1991)

```
Copyright (c) <year> <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of this
software and associated documentation files (the "Software"), to deal in the Software
without restriction, including without limitation the rights to use, copy, modify,
merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be included in all copies
or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF
CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE
OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

Example of an open-source permissive license: 3-clause BSD 3 (2009)

- **B**erkeley **S**oftware **D**istribution
- **Allows**: commercial use, modification, distribution, place warranty, **sublicensing**
- **Forbids**: use trademark, hold liable
- **Must**: include copyright, include license

https://opensource.org/licenses/BSD-3-Clause

Example of a proprietary license: **Zoom** (2022)

- **Zoom terms of service**
- *d **Prohibited Use**. You agree that **You will not use, and will not permit any End User to use, the Services to**: (i) **modify, disassemble, decompile, prepare derivative works of, reverse engineer or otherwise attempt to gain access to the source code of the Services;** (ii) knowingly or negligently use the Services in a way that abuses, interferes with, or disrupts Zoom's networks, Your accounts, or the Services; (iii) engage in activity that is illegal, fraudulent, false, or misleading, (iv) transmit through the Services any material that may infringe the intellectual property or other rights of third parties; (v) build or benchmark a competitive product or service, or copy any features, functions or graphics of the Services; or (vi) **use the Services to communicate any message** or material that is harassing, libelous, threatening, **obscene, indecent**, would violate the intellectual property rights of any party or is otherwise unlawful, that would give rise to civil liability, or that constitutes or encourages conduct that could constitute a criminal offense, under any applicable law or regulation; (vii) **upload or transmit any software**, Content or code that does or is intended to harm, disable, destroy or adversely affect performance of the Services in any way or which does or is intended to harm or extract information or data from other hardware, software or networks of Zoom or other users of Services; (viii) engage in any activity or use the Services in any manner that could damage, disable, overburden, impair or otherwise interfere with or disrupt the Services, or any servers or networks connected to the Services or Zoom's security systems. (ix) use the Services in violation of any Zoom policy or in a manner that violates applicable law, including but not limited to anti-spam, export control, privacy, and anti-terrorism laws and regulations and laws requiring the consent of subjects of audio and video recordings, and You agree that You are solely responsible for compliance with all such laws and regulations.*

https://explore.zoom.us/en/terms/

creative commons

Creative Commons licenses (2002)

*“Creative Commons licenses **give everyone** from individual creators to large institutions a standardized way to grant the **public permission** to **use their creative work** under **copyright law**”.*

Creative commons licenses

- Six types of **CC licenses**:
    - CC BY
    - CC BY-SA
    - CC BY-NC
    - CC BY-NC-SA
    - CC BY-ND
    - CC BY-NC-ND

BY (“Attribution”): credit must be given to the creator

SA (“ShareAlike”): adaptations must be shared under the same terms

NC (“NonCommercial”): only noncommercial uses of the work are permitted

ND (“NoDerivatives”): no derivatives or adaptations of the work are permitted

Creative commons licenses

- General licenses **for creators**
- Legal binding for **sharing**, **reproducing**, **diffusing** works

A question for you:
Why CC licenses exist if we already have FOSS licenses?

Creative commons licenses

- Are they the best option **for software**? **NO**
    - These licenses are **not designed for software specifically**
    - Instead, **use a specific software license for your software**
    - However, **OK for the documentation**

Dual licensing

Dual licensing

- Software **bonded to two or more different licenses**
- Example: Firefox tri-license (!): MPL 1.1, GPL 2.0, and LGPL 2.1
- **Is there a problem?** What about c**onflicting rights?** For example: obligation to release the source code, and possibility to close it in a proprietary product

Dual licensing

- **Yes, it's possible**. **You need to be the copyright holder**, of course

- Each license **grants users** some **rights**. They can **pick** the **license** they prefer
- **Two versions** of the same program**,** with independent development stories
- **For example:** one license can be **GPL**, the other **BSD**

Dual licensing: example

- Alice develops a program in the university
- She wants that it can be used in commercial products
- Alice licenses it under **BSD**: now a company can take her source code, and close it in a proprietary product **without asking for permission**
- Alice licenses it under **GPL**: her software now is free. All users benefit from being able to learn from the source code, they can use it and redistribute it. Some company would like to use it, but they can't since they'd have to release all their sources.
- Alice is the copyright holder of her software. Alice can thus put two licenses. One BSD for companies to use their code, and GPL for the community. **Dual licensing**.

Take-home messages

- A **license** is a **legal text** defining clearly what can and cannot be done with the source code
- **Free** and **open-source** software are **similar**. The **free software** movement focuses more on the **ethical aspects** and the **freedom** of the users. **Open source** more on the **convenience** of **sharing code** as a **development principle**.
- **Free software** is **not "*gratis*"**! It's about **freedom**. And it's totally **compatible** with **commercial** use.
- There are many **different** software **licenses**. You need to study and **pick** the **best** for the users and **for you**.

Software and patents

The idea of patents

- Patents go back medieval monarchs. Latin: *litterae patentes**
- **Idea**: give **exclusive permission** to someone to allow for making something. Allow for **technical advances**.

- **Example**: allow to make blades of steel for swords
- To arrive to the final product, the **inventor** dedicates **time and resources**, and experiments **until the result is satisfying**
- This allowed the **inventors** to **invest in their ideas without worrying** about **competitors**
- A **temporary monopoly** for the sake of a **great benefit to society**

* *open letters*, or an open document.

Software and patents

- **Software** is **based** on **logic**
- Similarly to **scientific research**, **contributions** are made **over previous contributions**
- *"Standing on the shoulders of giants"* (Bernard de Chartres?, Isaac Newton?)
- From the simplest foundational ideas (how to add two binary numbers?) to the most complex (how to implement a deep network with Keras?)

→ **Do patents for software make sense?**

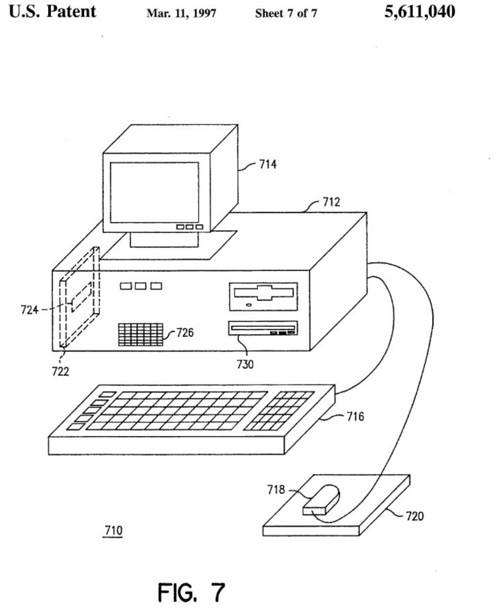
What do *you* think?

Current situation

- **European Patent Convention** (1973): **computer programs** are **excluded** from **patentability**
- **In France** are **excluded** “*les programmes ou séries d'instructions pour le déroulement des opérations d'une machine calculatrice*”.
- However, **laws are subject to interpretation**!
  - Schlumberger case (1981): it is acceptable to patent an invention that that was produced by a technical procedure where certain steps were implemented by software. See: https://www2.droit.parisdescartes.fr/warusfel/articles/JurInvLog_warusfel03.pdf
  - USA, June 19, 2014, the US Supreme Court (Alice Corp v. CLS Bank International case), rules that “*merely requiring generic computer implementation fails to transform [an] abstract idea into a patent-eligible invention*”

Some computer-related patents

- **Double click** (Microsoft, 2004). *Method and system for activating double click applications with a single click.* https://patents.google.com/patent/US5611040A/en



Some computer-related patents

Round corners in a tablet, and more, by Apple (2010): *Portable display device.* https://patentimages.storage.googleapis.com/72/b9/6d/e9f1850214cbcb/USD868775.pdf

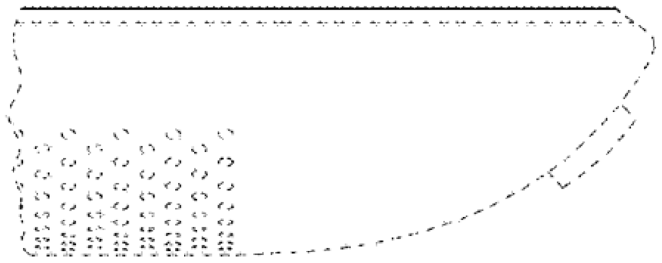


FIG. 10

Actually impossible to patent software? **No**

- **Laws subject to interpretability**
- Normally companies manage to patent software by considering **the system apparatus + software** as a **whole**.
    - Example, "*Automated determination of booking availability for user sourced accommodations*" (AirBnB, 2020) https://patentimages.storage.googleapis.com/97/27/84/418225e706573f/US20200019892A1.pdfFrom their text: "**Methods** and **systems** for updating a calendar entry for an accommodation listing are disclosed."

Why not software patents?

- **Discourage** software companies to produce **innovative software**
- **Discourage competitivity**.
  - **Large companies** try to **patent as much as possible**
  - **Patent wars**. **Patent *trolls***

- As a **small company** or **developer**, software patents most probably:
    - **Won't be possible** if **solely** related to **software**
    - If related to an **apparatus**, the patent **won't really protect** you in a **patent war**

Economic models of FOSS projects

Economic models

- A good **reference**: *The impact of open source software and hardware on technological independence, competitiveness and innovation in the EU economy.*
- **Freely available** here: https://op.europa.eu/en/publication-detail/-/publication/29effe73-2c2c-11ec-bd8e-01aa75ed71a1/language-en
- We'll briefly explore the **taxonomy** of **Okoli and Nguyen** (2016).

Economic models

- **Free** software is not ***gratis***
- However, you can **get the sources** and **compile** the programs **yourself**
- How is this **economically sustainable**?

Any ideas

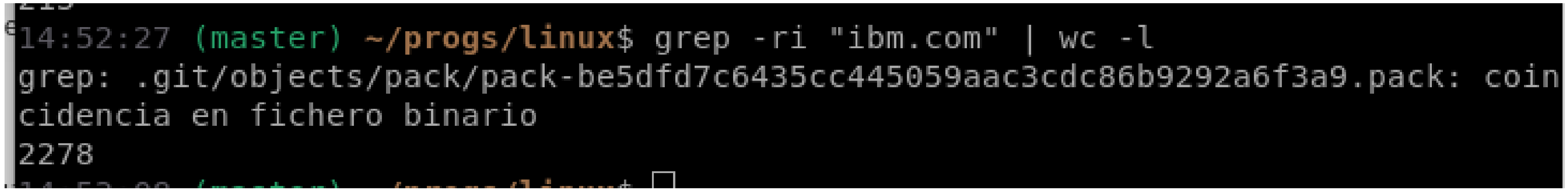
Economic models: **auxiliary services**

- Obtain **revenue** by offering **services around** the **software**, not directly by selling the software itself: customization, support, maintenance, **consulting**, localization, …
- Examples:
  - Google gives **Android** for free, including all development tools. It has also **Google Play**, where applications are sold, and Google obtains revenues from each sale.
  - Google gives **Tensorflow** for free. It also offers a complete and **paid cloud computing** infrastructure which is well adapted. Access to trained large neural networks.

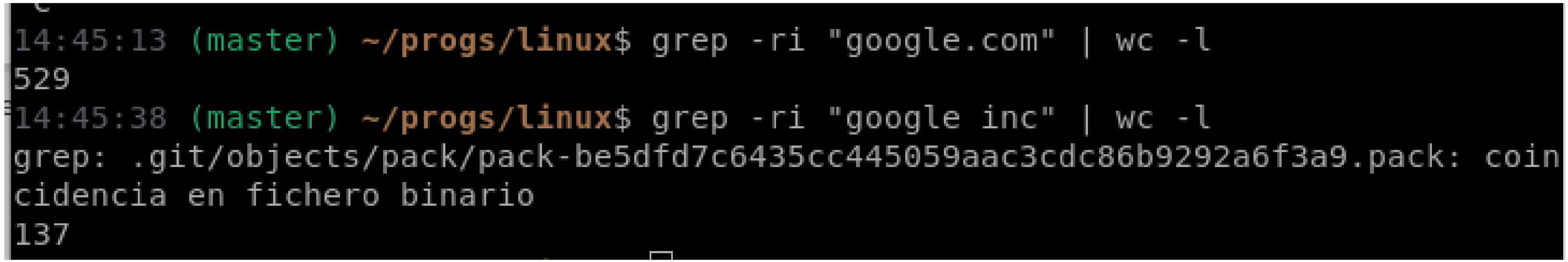
Economic models: **auxiliary services**

- Let’s try *grepping* the Linux source for contributions of companies...

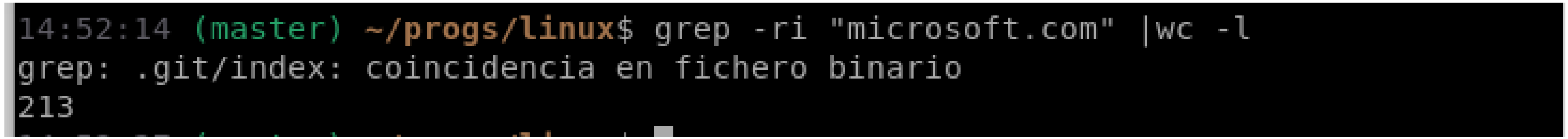
Economic models: **auxiliary services**



```
14:52:27 (master) ~/progs/linux$ grep -ri "ibm.com" | wc -l
grep: .git/objects/pack/pack-be5dfd7c6435cc445059aac3cdc86b9292a6f3a9.pack: coin
cidencia en fichero binario
2278
```



```
14:45:13 (master) ~/progs/linux$ grep -ri "google.com" | wc -l
529
14:45:38 (master) ~/progs/linux$ grep -ri "google inc" | wc -l
grep: .git/objects/pack/pack-be5dfd7c6435cc445059aac3cdc86b9292a6f3a9.pack: coin
cidencia en fichero binario
137
```

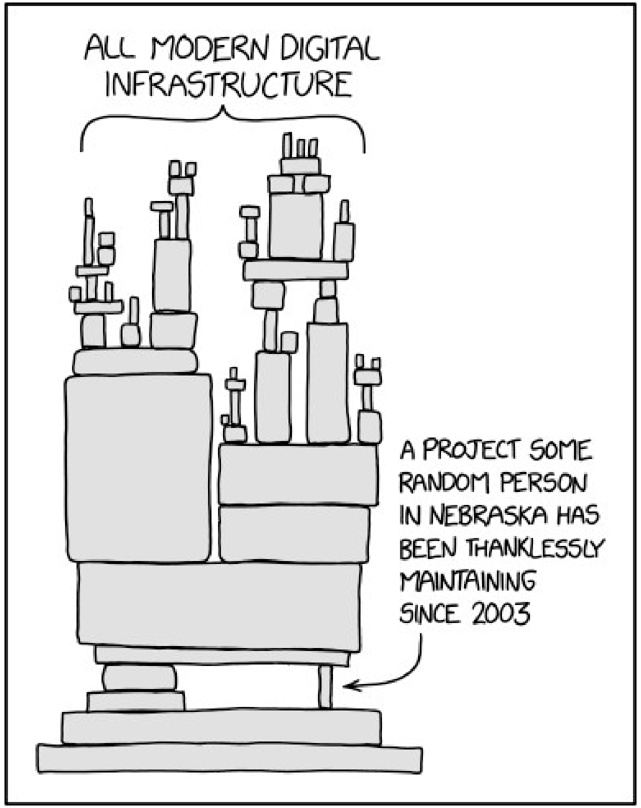


```
14:52:14 (master) ~/progs/linux$ grep -ri "microsoft.com" |wc -l
grep: .git/index: coincidencia en fichero binario
213
```

Economic models: **corporate development** and **distribution**

- **Organizations pay** the FOSS developers to ensure that the **product** is **maintained** and **customized** to their needs.
- Example:
  - *"Google funds two linux kernel developers to focus exclusively on security".* https://devstyler.io/blog/2021/02/25/google-funds-2-linux-kernel-developers-to-focus-exclusively-on-security/
  - IBM and others also fund the Linux kernel. And actively contribute to its source code.

- See: Log4j Zero-Day Vulnerability (CVE-2021-44228) Incident
- XKCD reference: https://xkcd.com/2347/


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Economic models: **software as a service** (SaaS)

- Running **software** is **complex**, since it requires: **compiling** it, a hardware **infrastructure** to run it (sometimes, a large distributed architecture or "the cloud"), **maintaining** it, …
- Instead, a company can sell the **paid product** to **run the service** in their **own system**, allowing for a **remote execution**.
- Related: Affero GPL license
- Examples:
  - Nextcloud for online file storage
  - Gitlab for git development
  - Wordpress blogs
  - …

Economic models: **dual licensing**

- The company decided to license the **product** under **two different licenses**:
  - A FOSS license. For example, GPL
  - A proprietary license, which doesn't need to be copyleft-compatible.
- Examples:
  - Oracle MySQL. GPL (https://www.mysql.com/products/community) + proprietary license. The proprietary products might evolve differently.
  - The R Project for Statistical Computing. GPL + proprietary licenses
  - Firefox with its 3 licenses

Economic models: **crowdfunding**

- The **community** is **interested** in having some product
- A team of developers proposes a **FOSS product**
- The team receives **donations** to build the product

- Examples:
  - Kickstarter campaigns during the development of the product. Minimal amount needed.
  - Patreon: recurring monthly payments to support the team of developers

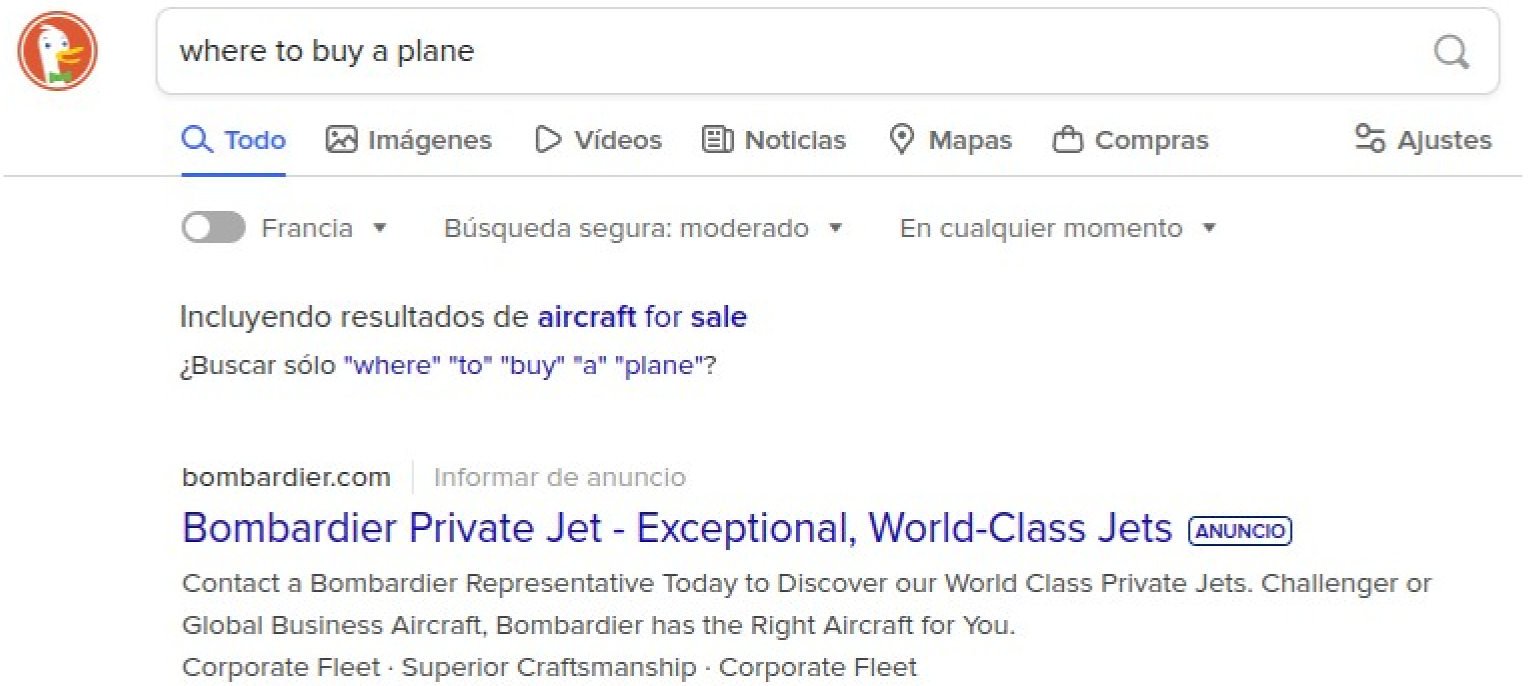
Economic models: **advertising**

- Typically in **SaaS**
- The **non-paid version** of the service shows **advertising**
- The **paid** version **doesn't**

- Examples:
  - Kickstarter campaigns during the development of the product. Minimal amount needed.
  - Patreon: recurring monthly payments to support the team of developers

Economic models: **advertising**

- Duck Duck Go search example



where to buy a plane

Francia ▾ Búsqueda segura: moderado ▾ En cualquier momento ▾

Incluyendo resultados de aircraft for sale

¿Buscar sólo "where" "to" "buy" "a" "plane"?

bombardier.com | Informar de anuncio

Bombardier Private Jet - Exceptional, World-Class Jets ANUNCIO

Contact a Bombardier Representative Today to Discover our World Class Private Jets. Challenger or Global Business Aircraft, Bombardier has the Right Aircraft for You.

Corporate Fleet · Superior Craftsmanship · Corporate Fleet

Case studies

Case studies

- **Complete free GNU/Linux systems**: GNU/Linux distributions

- **For scientific research**: scikit-learn, matplotlib, numpy, scipy, Tensorflow, Pytorch, Bioconductor, FFTW, Octave, Jupyter, IPOL, ...

- **Widely used:** LaTeX, Firefox, Chromium, LibreOffice, GIMP, VLC, Linux, Blender, GNU C/C++ compilers, Python, Thunderbird, …

Case studies

- 9/10/2024: **Nobel prize** to **David Baker**, **Demis Hassabis**, and **John Jumper** in **Chemistry** to reveal computationally the structure of proteins
- With **AlphaFold2**

- **Apache 2 license**
  https://github.com/google-deepmind/alphafold/blob/main/LICENSE