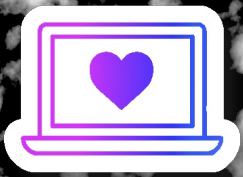
Libre!

The zine on free software for young people

Issue 1
February 2026

In this issue:



Made using all free software. No AI was used in making this magazine.

For corrections, comments and ad sales, please contact the author at carmen@missionlibre.org.

About this zine

This zine is a publication of Mission:Libre, a new project for young people who care about free software -- software everyone can share, study, improve and change.

Learn more about what Mission:Libre is doing for young people in free software and how you can help at https://missionlibre.org

Mission:Libre

Hello!

Happy New Year!

We're kicking off 2026 with some information about that most important of issues: privacy. I'll tell you why privacy matters and what's at stake if you lose it. You'll also find a handy guide to two of the most serious ways your privacy is being taken from you: surveillance by government spy agencies and tracking by online advertisers.

In the second half, we get *close* to the machine. First, we'll have a look at number systems. You'll learn how binary works and why F is a number. After, we'll use our new knowledge to find out how to speak the language your computer actually knows: not Python, not JavaScript -- machine code!

- Carmen-Lisandrette

68 65 65 6C 6F
77 6F 72 6C 64

Your privacy matters

Each time you use a computer or a phone, you create thousands of little bits of information that can tell someone all about you. Many companies, governments and organisations want this data -- yes, even yours!

Let's find out who they are and why you should care.

Who wants your data

Governments

Governments collect a lot of information in the name of preventing terrorist attacks. The NSA programmes talked about in the next article were started for this reason.

There are other reasons governments want data about you, not all of which are public. Your data can be useful for deciding government policy or enforcing laws, for example.

Political parties and people who want to persuade you

Information about what people buy, read, where they go or who they spend time with can be useful if you want to tailor your message to someone.

Most political campaigns, and some larger non-profits, purchase lots of information on people they can use to guess at how they might vote.

online advertisers

Data about how you use the internet is used to choose which ads you see online. Companies also want this information so they can figure out what you'll buy. Advertisers can even narrow down the people who see an ad to the few people they know are have the power to vote on something or make a big purchase.

...and more!

There are some other reasons companies might want your data.

For example, Amazon Alexa collects snippets of what you say around it and sends it away to make Alexa better at understanding you. A company that writes software might collect information on what you do with their programs so they can figure out how to keep you using it for longer.

Why you should be worried

There are thousands of different uses for data. Unfortunately, not all of them are good for us.

Sometimes people shrug and say they aren't embarrassed about what's out there about them. But privacy isn't just about being ashamed. It's about our right to be treated fairly, our safety and our ability to make choices freely.

Our data tells people...

WHAT MAKES US HURT.

What we secretly fear and hope for.

What will make us change.

So they can...

MANIPULATE HOW WE FEEL.

Nudge us to spend time and money on things we wouldn't want to otherwise.

Influence us into supporting causes that aren't in our best interest.

If we find ourselves on the wrong end of government power, our data can be used against us

In the US, ICE has bought data from the same sources marketers use to help them imprison and deport immigrants[1].

Surveillance also makes people afraid of challenging their governments and being curious about what's going on in the world. When newspapers reported on the American government spying on the internet, many people stopped visiting Wikipedia pages on terrorism and other senstive topics[2].

Our data can be stolen

Wherever there's lots of data, there's someone ready to steal it.

Last October, data from Salesforce, which makes software companies use to record information about their customers, was stolen and put online. Leaks like this happen every day.

Our data can be used by people who want to harm us

The same data that helps a company convince us to buy more bread can also be used as blackmail or can help scammers fool us.

Learn more:

[1]"ICE Buying American's Location Data Under Scrutiny": https://www.newsweek.com/ice-buying-americans-location-data-under-scrutiny-11627381

[2]"New Study Shows Mass Surveillance Breeds Meekness, Fear and Self-Censorship": https://theintercept.com/2016/04/28/new-study-shows-mass-surveillance-breeds-meekness-fear-and-self-censorship/

Who's spying on you?

It might be uncomfortable to think about, but finding out who might be invading your privacy you can help you make choices to protect yourself.

We're going to look at two of the most common ways people get information from what you do on your devices. The first is surveillance by government spy agencies. The other is tracking in websites and apps.

Government Surveillance

It might sound like something from a movie, but the spy agencies of the US, UK, Canada, Australia and New Zealand are watching what everyone does online.

Many people used to think this was just a conspiracy theory. Sure, there had always been rumours that there was stuff going on in secret. But it couldn't have affected *ordinary* people, right?

Well, it wasn't until Edward Snowden shared some secret documents from America's National Security Agency that people found out just how much spying was really going on.

In 2013, The Guardian, a UK newspaper, put out the first article based on the Snowden files: "NSA collecting phone records of millions of Verizon customers daily". In the following months, other newspapers published more. They all added up to a huge conspiracy between the US, UK, Canada, Australia and New Zealand ("Five Eyes") to spy on each others' citizens. This is what we found out:

The NSA and the FBI forced the internet's biggest companies to let them into their systems. Some of the companies were Google, Facebook, Microsoft, Apple, Yahoo! and Skype. This means the NSA and FBI could get e-mails, files "in the cloud" and phone calls. This programme is called PRISM.

The GCHQ (like the UK's version of the NSA), can read what flows through the big cables that carry the internet around the world. It keeps a lot of this information without anyone else knowing.

The NSA records lots of information about the phone calls made by the millions of people who use Verizon. This included what numbers were called, how long people talked for and what time the calls were made.

A computer program called XKeyscore made it easy for spies to search what anyone e-mailed, searched or browsed so long as they had the person's e-mail address. This was possible because back then most things on the internet were sent in a way that let anyone read them.

This shocked people. The rumours, and more, were true.

Things have gotten better, a little. Now, people make sure to send most information over the internet in a way others can't read.

However, we don't know what the NSA is doing today. The NSA is famously paranoid and the courts which are mean to protect everyone are secretive. People who have worked in those courts have said they are very willing to give permission to spies to do almost anything they want. It's impossible for the average person to know what the NSA is doing and if they're following the law. We should assume mass surveillance is still going on.

Another problem is metadata. Metadata is data about data. It's things like the addresses of pages you view, the account names you message, and the times you do things. Even though metadata isn't the actual contents of what people are doing, it still says a lot. For example, anyone who knows you downloaded *Libre!* can guess you have an interest in free software and privacy.

Online Tracking

It's not just spies that want to know all about you. There are many companies online who's business is collecting, analysing and even selling what you do online. This is how they do it:

Cookies. Cookies are small text files websites put on your computer which sites can ask it to send back. They can work like a label unique to you. If a site puts an ad company's cookie on your computer, they can connect the dots and find out

Browser Fingerprinting. Instead of using a cookie, advertising companies can figure out which computer is visiting a site in a different way. Your computer is unique like a fingerprint: only you will be using the exact same programs and the same settings.

Large platforms don't need cookies or browser fingerprinting to find out all about you. They can figure it out based on what you like, watch, read or listen to. The information they have on you might be used to make you spend more time on the app or advertise to you. For example, spotify guesses your mood by how you're listening and uses it choose what ads it will show you (see https://thebaffler.com/downstream/big-mood-machine-pelly for more information). Creepy, right?

How Free Software Helps

Free software can help you keep your online life private. Free software is much less likely to sell your information or use it to advertise to you. It's also less likely to trick you into making choices that are bad for your privacy.

This is because harmful features can be removed from free software by others who aren't the developer. In this way, helpful people can keep us safe.

But you should be aware that free software can't give you perfect privacy. Free software can't help you against programmes like PRISM that collected data long after it left people's computers. It also can't protect you against the other ways your data is collected, like the companies that sell your purchase data from their stores or the number plate readers on the motorway that might follow your car around town.

In the next issue we cover some simple ways to win back your privacy. Keep an eye out!

Count like a computer

Binary and Hexidecimal numbers

Ten fingers, ten digits

When you were in primary or elementary school, you probably had a lesson on place value. Your teachers probably showed you a chart that looks like this:

hundred	ten	one
3	4	2

$300 + 40 + 2 = 342$

What are exponents?

An exponent tells you to multiply a number by itself. The big number (the *base*) tells you what you should multiply, the little number (the *power*) tells you how many times.

10^2 means 10x10
10^3 means 10x10x10
and so on.

You should also know there are two special cases:

$10^1 = 10$
$10^0 = 1$ (anything to the power of 0 = 1)

Let's see what happens when you rewrite the heading of the chart using exponents:

10^2	10^1	10^0
4	7	2

$4 \times 10^2 = 400$
$7 \times 10^1 = 70$
$2 \times 10^0 = 2$

$400 + 70 + 2 = 472$

It works for bigger numbers, too:

10^4	10^3	10^2	10^1	10^0
7	3	2	6	2

$10^4 \times 7 = 70000$
$10^3 \times 3 = 3000$
$10^2 \times 2 = 200$
$10^1 \times 6 = 60$
$10^0 \times 2 = 2$

Clearly our system is based on powers of ten, Because of this, we say it's *base-10* or *decimal*. Decimal comes from the Latin word for ten.

Binary

A long time ago, people tried to use the decimal system to build computers. The computers they came up with were very unreliable and expensive.

It turns out it's much easier to build a computer if you use only two values: on and off, true and false, 1 and 0. That means that computers have to use a number system inside that only has two digits. That's the *binary* or *base-2* system.

The binary system works like the decimal system, but with powers of two instead of 10:

2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	1	0	1

$32 + 0 + 8 + 4 + 0 + 1 = 45$

$2^5 \times 1 = 32$ $(2^5 = 32)$
$2^4 \times 0 = 0$ $(2^4 = 16)$
$2^3 \times 1 = 8$ $(2^3 = 8)$
$2^2 \times 1 = 4$ $(2^2 = 4)$
$2^1 \times 0 = 0$ $(2^1 = 2)$
$2^0 \times 1 = 1$ $(2^0 = 1)$

Try it yourself! Using the list of powers of two below, see if you can convert these numbers to binary. Answers are in the back:

a) 8195 b) 38945 c) 300 d) 1000

Q: Why do numbers like 4096, 1024 and 32,768 come up so much in computery stuff?

A: Because they're powers of two

$2^0 = 1 = 1$
$2^1 = 2 = 10$
$2^2 = 4 = 100$
$2^3 = 8 = 1000$
$2^4 = 16 = 10\ 000$
$2^5 = 32 = 100\ 000$
$2^6 = 64 = 1\ 000\ 000$
$2^7 = 128 = 10\ 000\ 000$
$2^8 = 256 = 100\ 000\ 000$
$2^9 = 512 = 1\ 000\ 000\ 000$
$2^{10} = 1024 = 10\ 000\ 000\ 000$
$2^{11} = 2048 = 100\ 000\ 000\ 000$
$2^{12} = 4096 = 1\ 000\ 000\ 000\ 000$
$2^{13} = 8192 = 10\ 000\ 000\ 000\ 000$
$2^{14} = 16384 = 100\ 000\ 000\ 000\ 000$
$2^{15} = 32768 = 1\ 000\ 000\ 000\ 000\ 000$

Hexadecimal

Binary numbers are confusing for people to work with. A shorter way of writing them is to group the digits into fours and give them each a digit or a letter.

```
0100 1001 1101 0101
4    9    D    5
```

(You'll often see numbers like this written as 0x49D5. The 0x part tells you what's after isn't a decimal number.)

```
0001 = 1
0010 = 2
0011 = 3
0100 = 4
0101 = 5
0110 = 6
0111 = 7
1000 = 8
1001 = 9
1010 = A
1011 = B
1100 = C
1101 = D
1110 = E
1111 = F
```

It turns out this is exactly the same as using a base-16 number system! We call that system *hexadecimal*. Hexa is the Greek for 6, so hexadecimal really means '6-10 system'.

This is what the place value chart for the hexadecimal system looks like:

16^3	16^2	16^1	16^0
2	A	3	1

$16^3 \times 2 = 4096 \times 2 = 8192$
$16^2 \times A = 256 \times 10 = 2560$
$16^1 \times 3 = 16 \times 3 = 48$
$16^0 \times 1 = 1 \times 1 = 1$

$8192 + 2560 + 48 + 1 = 10801$

When we write a hexidecimal number, we need extra digits to stand for numbers that are bigger than decimal 10. A to F are digits! A stands for decimal 10 and F stands for decimal 15.

Let's give it a go! Try to convert these numbers into hexadecimal. Again, answers are in the back:

a) 40960 b) 12320 c) 43981

A Challenge

Some very old books didn't use hexadecimal. Instead, they used a number system called octal.

Using a dictionary, look up the prefix *oct-*. Can you figure out how octal works?

See if you've gotten in right by changing these octal numbers to decimal. Answers are in the back cover:

a) 4763 b) 37632 c) 142 d) 14

Bonus question: is there an easy way to convert octal to binary and back again? Hint: Read the first two boxes on hexidecimal again.

Talking in 1s and 0s

How machine language works

What is machine language, anyway?

When we write programs, we write them in languages that made so people find them easy to read and write. Computers can't run the code we write unless we translate it to machine code. Let's look at an example.

To the right is a program in C. It doesn't do much. It just adds two numbers and saves them in the computer's memory.

```
int main() {
      int x = 3;
      int y = 6;
      int answer;

      answer = x + y;
}
```

My compiler gave me this assembly code on the right. (If you don't know, a compiler is the program that translates the code you write into something the computer can run).

Assembly is closer to machine code, but it's written with letters and numbers.

```
movl    $3, -12(%rbp)
movl    $6, -8(%rbp)
movl    -12(%rbp), %edx
movl    -8(%rbp), %eax
addl     %edx, %eax
movl    %eax, -4(%rbp)
```

Your computer can't run assembly, either. To show you what assembly looks like I asked my compiler to stop half-way through its work. If I let it go all the way through, it gives me this:

```
c7 45 f4 03 00 00 00      movl   $0x3,-0xc(%rbp)
c7 45 f8 06 00 00 00      movl   $0x6,-0x8(%rbp)
8b 55 f4                  mov    -0xc(%rbp),%edx
8b 45 f8                  mov    -0x8(%rbp),%eax
01 d0                     add    %edx,%eax
89 45 fc                  mov    %eax,-0x4(%rbp)
```

The bolded part is the machine code. In the computer, this is all binary, of course.

Each of those lines is an instruction that does a tiny bit of work. The first two put the numbers 3 and 6 into memory. The next two move the numbers so the computer can add them in the line after. The last instruction saves the answer to memory.

The CPU

The CPU is the actual part of your computer (or phone) which understands the code that's in your programs. It's plugged into your computer's motherboard, which connects it to other parts of your computer like the main memory and disks.

This is what a CPU looks like!

Registers

Registers are a small amount of memory in the CPU itself. There are only a few registers: an Intel processor like you might have in a computer has 16 of them, which all together hold 128 bytes (a byte is 8 binary digits). That's nothing compared to main memory: a computer with 8GB of RAM might have 8589934592 bytes of it!

Registers are an extremely fast form of memory. Because they're so fast, they're used for data that's we're using right away, like the numbers we added in the program before. Other bits of data, and the program itself, are kept in main memory.

Fetch - Decode - Execute

The CPU goes through three steps when it's running each instruction:

Fetch – The computer looks in the program counter, which is one of the registers used for keeping the address in main memory where the next instruction is. Then it collects the instruction for the next step.

Decode – The CPU figures out what the instruction is saying.

Execute – The CPU does what the instruction says.

A CPU can do all of these steps millions of times a second!

Let's write a (very small) assembly program!

Let's write a very simple assembly language program. All it's going to do is copy two numbers into registers, add them and save the result.

The instructions we'll need are `mov`, `add` and `push`. `add`, well, adds. `push` copies a register to memory. `mov` can be used to move information around both the registers and memory.

We'll keep the numbers we want to add in the registers `rax` and `rbx`.

First, let's write our code in assembly language. Lines in assembly language go like this:

```
mov %rax,%rbx
```

The first part is the instruction we want to use. The other parts are the information we want it to work on. In this case, we're telling mov to copy the contents of `rax` to `rbx`. The `%` means we're talking about a register, not a place in main memory.

Our first two line in our program will put the numbers we want in the registers so we can add them. I'm choosing to add the hexidecimal numbers 7 and 80. `$` means I really do

want to say the number, not a place in memory or a register.

```
mov $7,%rax
mov $80,%rbx
```

Onto the next line. The `add` instruction will add what's in `rax` to `rbx`, and put the answer in the last one.

```
add %rax,%rbx
```

Just one more. This line will copy the contents of `rbx` to main memory. The computer already knows where we want to put it.

```
push %rbx
```

All together, our program is:

```
mov $7,%rax
mov $80,%rbx
add %rax,%rbx
push %rbx
```

Easy enough, right?

The machine code

Assembly language is meant for people. If we want something the computer could run, we still have to convert it to machine code.

Machine code is very hard to write, so I used a program to translate it into machine code. This is what it gave me:

```
48 c7 c0 07 00 00 00      mov   $0x7,%rax
48 c7 c3 50 00 00 00      mov   $0x50,%rbx
48 01 c3                  add   %rax,%rbx
53                        push  %rbx
```

Each pair of numbers is a byte. For example, the first line of machine code looks like this in binary:

```
0100 1000 0000 0001 1100 0011
   48        01        c3
```

Look at the second line. See the bit going 48 01 c3? That's the actual instruction for the computer. Those three bytes mean "copy the four bytes after this instruction into the register rbx". You'll see the number 50 from our program. Yes, the computer I use writes numbers backwards!

That's the end of our introduction to machine language -- and this edition of *Libre!* I hope you've had fun and learnt a lot.

Look out next time for more, including guides to preserving your privacy and some basic Python!

As always, you can e-mail me at carmen@missionlibre.org.

See you again soon!

Acknowledgements

Cookie icon by EmojiOne CC-BY-SA Intl. 4.0 https://creativecommons.org/licenses/by-sa/4.0/deed.en

Phone icon from by Deepin (Wuhan Deeping Technology Co., Ltd.). This work is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or any later version. This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See version 3 of the GNU General Public License for more details (https://www.gnu.org/licenses/gpl-3.0.html)

CPU photo by Rjluna2 on Wikimedia Commons. CC-BY-SA Intl. 4.0

Answers to questions in "Count Like a Computer"

Binary:

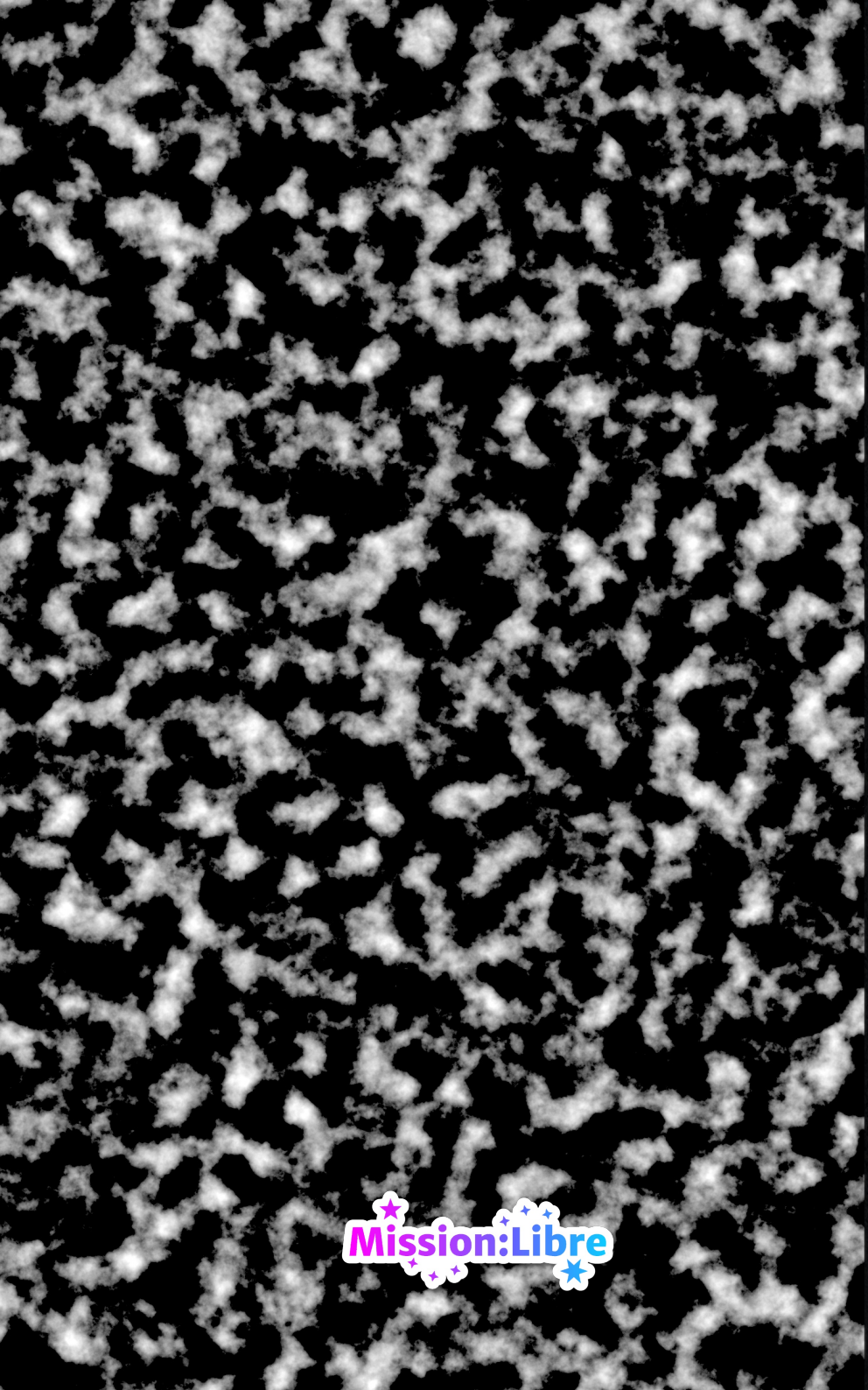
a) 10 0000 0000 0011
b) 1001 1000 0010 001
c) 1 0010 1100
d) 11 1110 1000

Hexadecimal:
a) A000
b) 3020
c) ABCD

Octal Challenge:
a) 2547
b) 16282
c) 98
d) 12

Bonus question: Instead of taking bits in 4s like hexadecimal, octal takes bits in 3s:

```
100  101  110  111
 4    5    6    7
```


Mission:Libre