Mika Motskobili

LEVER BURNS

_____ no .....^c

Thesis submitted to: the Department of [Experimental Publishing or Lens-Based Media],
Piet Zwart Institute, Willem de Kooning Academy,
in partial fulfilment of the requirements for the final examination for the degree of:
Master of Arts in Fine Art & Design: [Experimental Publishing or Lens-Based Media].
Adviser: Marloes de Valk
Second Reader: Manetta Berends
Word count: 7245

|| _ ____ [$

I'm writing this on my web server terminal, using one of my favorite commands -- xargs.
In its stripped-down form [i.e. with no arguments, but only -0 option] it allows me to write new lines without command-line interface [cli] interpreting them as executable commands.
But when I hit Enter it goes to a new line, after which I can't go back and edit it in this environment / session -- no backspaces in here.
Ctrl+d exits the command and logs the input into the text file: I'm funneling this session into ch00.txt.

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

$ xargs -0 >> /var/www/kibkadar/the/ch00.txt

..............................................................
```

http://leverburns.blue/the/ch00.txt

This, of course, impacts the writing itself: once you are on a new line you just need to move forward and leave revising for later.
This method can push the process in two opposite modes: you may consider your thoughts cautiously before logging them or you are gonna jot them down like there's no tomorrow.
The former is one of the reasons I chose this tool.
I slowed down and thought the sentences through before typing them.
I used other methods and tools as well and I've noticed that parts that are written in xargs are the least edited ones later on.
Another reason for picking xargs, and terminal in general, is because I enjoy this environment and it has seeped into and permeated my daily life at this point.

The terminal or a command-line interface [cli] is an application that deals with a text input / output.
The commands typed in a terminal are processed by a command-line interpreter [shell] and the output is fed back into the cli.
Shell is a command-line programming language -- a layer between a user and a kernel [a principal computer program in charge of a software-hardware interaction of the system] that communicates the commands to the operating system.

> :// There are various types of shell programs: csh, tcsh, ksh, zsh, bash etc. Bash [Bourne Again Shell] is the most widely known and is available as a default option on many operating systems. It is an open-source* and advanced successor of Bourne shell [sh], developed in 1989 by Brian J. Fox in a clean room** to avoid a copyright infringement (Yitbarek, 2019).
>
> * Open-source software [OSS] denotes a type of application whose source code is publicly available and it's up to the users to use, modify and distribute it however they see fit. Despite having its pitfalls, such as being heavily dependent on a volunteer work, which may lead to some instability, this approach implies a great potential for developing

an application faster and/or in more interesting ways on a larger collaborative scale than in a scenario where only handful of people ruminate on it. In Python's Tale (2019) Diane Mueller emphasizes the importance of a community in relation to the open-source practices:

> ---
> 'So that idea, that contribution isn't just about code, it's about participation, it's about learning and education, and it's a lot about documentation, was the way into community for a lot of people.'
>
> ---

Open-source and free software are tightly connected concepts and an application that qualifies as both is referred to as FOSS [Free and open-source software]. Free software source code is also released under the license which allows its end-users liberties similar to the ones of the OSS, but at its core, there is a strong emphasis on the users and their right to be in charge of their copies of the software.
In a conversation with Cornelia Sollfrank (2018), spideralex recalls Laurence Rassel's clear-cut response to the question of 'why is free software feminist':

> ---
> 'Operating system in French is "système d’exploitation" [exploitation system]. And she said, as a feminist, what you want is to access your exploitation system, and to be able to change and modify it.'
>
> ---

Spideralex then proceeds with emphasizing the significance of not only being in charge of our tools and devices but permeating all underlying and connected layers of technology with the same attitude since they are also drenched with 'a lot of binary, capitalist, patriarchal code' all the way through.

** Clean room refers to a practice of reverse engineering without accessing a source code / material.

Through the terminal you can have access to your machine's operating system, run programs, create and modify files / scripts, and break things.
Angelica Blevins and Zach Mandeville put it poetically in their 'the map is the territory' (2020) zine:

> ---
> 'This is a place of empowerment, tangible creativity, and mystic bewilderment.
> While it can be dangerous, it’s also exceedingly helpful if you know how to listen.'
>
> ---

It is a powerful tool and more importantly it is available on every operating system by default.
Although, some commands may be OS-specific.
For instance, to install software packages on Linux you run apt-get install <software> command, while the MacOS equivalent would be brew install <software>.

> :// In comparison, users operating within the Graphical User Interface [GUI], which is a commonplace scenario, would go to a software distributor web-store [Microsoft Store, App Store etc.], download an application and follow an installation guide inside its interface.

Commands can be as simple as echo, which returns back the string you give it.

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

$ echo no
no
..............................................................
```

xargs is a more elaborate tool, in that it can receive an output of a command and stream it into another command as an input

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

$ echo {0..7} | xargs touch

..............................................................
```

Here, xargs takes the output of echo {0..7} [numbers from 1 to 7] and pipes it into touch [used to generate files], which then creates empty files titled 1-7 in a current directory*.

* 'Directory' and 'folder' are frequently used interchangeably.

I personally find using a terminal very fun.

I can't fully explain this though, nor do I think that everything needs to or can be explained, not through words anyway.
The fact is I feel comfortable in this space and the terminal made it easier for me to start writing this text.

:// Alternative and a common [default] way of navigating and interacting with your computer is of course the Graphical User Interface [GUI] -- a set of visual representations of software functionalities, which allow user interaction by clicking on the icons, or text input in a search bar etc.

As Johanna Drucker points out in Reading Interface (2013), the GUI is in no way a representation of underlying computational processes through which an action happens when clicking on the icons, but it is a 'boundary space' where these processes are abstracted through metaphors. It's not on everyone's mind or nor can they afford the time to go beyond this carefully assembled visual veneer. I used to check every corner of the operating system through the GUI, just to know how things were arranged by default or how I could modify it, but there is only so far a Graphical User Interface can take you; that's how I ended up here -- behind the scenes.

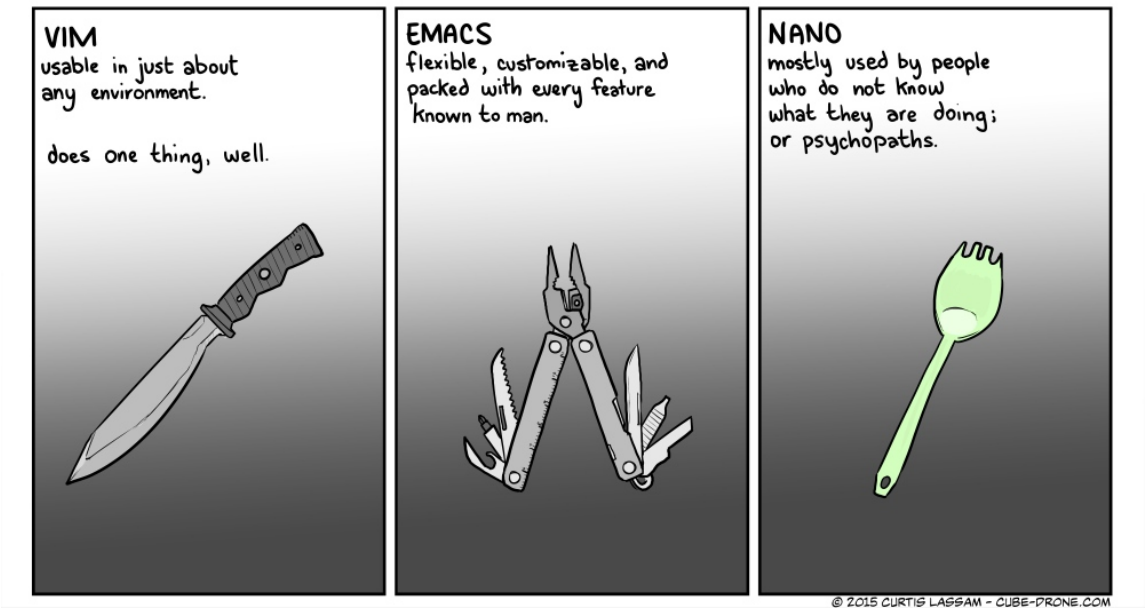
:// During their talk at LibrePlanet Conference (2021), Tech Learning Collective [TLC] stressed the importance of understanding the metaphors in order to have a better grasp on 'what else you can do with computers' or even devise your own metaphors to that end. I find this approach very welcoming and accessible for the regular users, as it doesn't require ingesting extra technical knowledge, but rather gives room for considering the metaphors more extensively and carefully, and possibly appropriating the interfaces you interact with on a daily basis.

> __ _ In Kochuu (2003), a documentary film about Japanese architecture, Kisho Kurokawa points out how representation of mountains or rivers through sand and gravel in Japanese gardens are 'much more enriching than having real water there'. To me this thought links back to, however rigidly, the Tech Learning Collective's approach to the metaphors; even though the graphically denoted folder doesn't stand in for an actual folder, functionality-wise, this image can be impactful nonetheless and especially so, if one attaches an expanded / modified meaning to it.

:// In 1999 Ted Nelson refers to these metaphors as 'scraps of resemblance that tie us down', and argues that 'software should instead be designed without regard to past resemblances, with independent conceptual structure that may take any appropriate shape'. With this, Ted [being Ted] challenges the entire idea of the HOWs and WHYs of the interfaces and considers emulating real-world objects / concepts within the computer systems to be a mistake and a missed opportunity. It's hard to imagine what usership, interfaces and computational systems in general would look like had this approach been implemented from the get-go, but it's interesting

to have this point in mind moving forward.
It's important to make a distinction regarding what Nelson and TLC are addressing: Ted Nelson refers to the ideas behind a software design, while TLC is concerned with making already existing tools useful for communities and individuals in their everyday lives.

Using cli predominantly means that I don't have to spend time on familiarizing myself with new interfaces with every new application.
Although there are recurring elements when dealing with a command-line interface [certain flags, redirection [>>], piping etc.], some cases are trickier and takes getting used to. For instance, command-line text editors Vim, Nano and Emacs are quite distinct.



Moreover, running the programs in a command-line interface takes less processing power, i.e. it's lighter on your computer.

Here is what happens in a background when I launch the LibreOffice Writer and gedit [text editor applications], and open nano in a terminal [$ sudo nano x.txt]:

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

$ ps -eo %cpu,%mem,vsz,rss,command | awk 'NR==1 || /libreoffice/ || /nano/ || /gedit/'

%CPU %MEM    VSZ    RSS COMMAND
 0.0  0.0   8756   3144 nano x.txt
 0.0  0.0   175836 5724 /usr/lib/libreoffice/program/oosplash --writer
 23.7 1.8   973740 300156 /usr/lib/libreoffice/program/soffice.bin --writer --splash-pipe=5
 1.3  0.5   953764 83824 /usr/bin/gedit --gapplication-service
 ..............................................................
```

ps aux command would print out the list of all the current processes [running programs] onto the terminal with more columns of data, however ps -eo %cpu, %mem,vsz,rss,command narrows down the query to the CPU, MEM, VSZ, RSS and COMMAND

columns and awk 'NR==1 || /libreoffice/ || /nano/ || /gedit/' grabs the first line of the output [NR==1], which happens to be a header [denoting the names of the columns], and the lines related to the LibreOffice, gedit and nano processes [/libreoffice/ || /nano/ || /gedit/].
__ %CPU column contains the information regarding the percentage of the Central Processing Unit that is being used to run the particular application. As the command output shows, upon the launch, Libreoffice application [soffice.bin in particular] accounted for 23.7% load [this number drops eventually], gedit was less taxing [1.3 %], while nano was exquisitely light [0%]
__ %MEM displays the percentage of the RAM memory currently used by the process. This column also indicates that Libreoffice and gedit text editor are using up more space than nano.
__ VSZ [Virtual Set Size] denotes a memory size allocated to a process.
__ RSS [Resident Set Size] shows how much space [in kilobytes] of the RAM the task is occupying.
There are more ways to monitor the active processes on your machine: running glances command on a terminal or launching the System Monitor application etc.

Notably, cli can also be a serendipitous space, which is even more exciting.
I have stumbled upon some functionalities purely by accident of typos or 'wrong' keyboard shortcuts.

Additionally, the command-line interface affords me to avoid some third-party proprietary* applications.
I have done all my video editing only on the terminal with FFmpeg, for instance.
I haven't even checked if my operating system comes with a video editing software.
I guess we'll never know…

* Proprietary [non-free / commercial / closed-source] software refers to a copyrighted application. The publisher outlines a scope of the rights [in an end-user license agreement [EULA] or terms of service agreement [TOS] etc.] users can have over the software. This may entail the limitations regarding the access to and modification of a source code, and further its distribution. The owners publishing the software under this license are primarily leasing the rights of using the application. iMovie [proprietary video editing software by Apple Inc.] and FFmpeg [free and open-source software for video/audio processing, originally developed by Fabrice Bellard] are two straightforward examples of applications with opposite types of licenses.

:// On a more dramatic and harmful side of using a proprietary software / system stands the relatively recent development with Apple: on the macOS 11 Big Sur release day in November 2020 all non-Apple applications running on the internet-connected mac computers crashed (Paul, 2020). The company checks the app certificates status through OCSP [Online Certificate Status Protocol], which is implemented on ocsp.apple.com servers (Engst, 2020). At the same time Apple logs the data [such as Date, Time, Computer, ISP, City, State, Application Hash] of the users' machines whenever they are launching the programs and at some point, when the ocsp.apple.com servers slowed down, this process glitched which lead to the application failures. This is a scenario where you, as a user don't have much say in how your machine operates or what happens to your data.

You can't be in control of a software in a meaningful way while you don't have access to its mechanics.

:// On the other hand, there is the case of youtube-dl, an open-source command-line software for downloading videos from Youtube, Vimeo, and many other video-hosting platforms: in October 2020 GitHub* removed the youtube-dl repository** in response to the take-down notice by Recording Industry Association of America [RIAA] claiming that the software was violating the Digital Millennium Copyright Act [DMCA] (Harmon, 2020). Fortunately, youtube-dl soon was restored on GitHub since RIAA allegations were rebutted (Harmon and Stoltz, 2020). So, even though the tool itself was published under Unlicense*** license (Valsorda, 2012), it still came under legal scrutiny since it had the potential to deal with the copyrighted materials hosted on Youtube. This was a complicated encounter between open-source and proprietary agents on a Github playground.

* GitHub is a platform for hosting and collaboratively developing the code.
** Git repository is a directory where the project / code is stored on GitHub.
*** Software under this license belongs to the public domain by default.

I edit this text either in nano [a command-line text editor] or on a pad [of the Etherpad application].
With nano I remain in a terminal environment, although, I only did some minor text alterations here.

||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

```
$ sudo nano /var/www/kibkadar/the/ch00.txt
```

.....

Etherpad is an open-source collaborative writing web application.
I'm self-hosting an Etherpad instance on this server.

http://eth.leverburns.blue

_____

Previously I would use the XPUB-hosted Etherpad service [https://pad.xpub.nl/].
Self-hosting it has shifted my relationship with this tool in a significant way: I feel an added layer of intimacy, probably because I know where the data is stored, only I have access to the list of all the pads [I am also able to delete them], and I modified the interface quite a bit and expanded the functionalities by installing the plugins.
Through this process, I got to know its overall architecture and source code, meddled with it sometimes and ended up being upset occasionally.

You are very much invited to use it, but please keep in mind that the application might not always be available for a variety of reasons: the server might crash or it could be unplugged due to relocation, or this Etherpad instance itself may go down or be out of reach because I'm messing with it [sometimes it goes terribly, fatally wrong].

So, I would suggest downloading the pads on your computer just in case.

Some parts of this text are transcriptions of my speaking.
This way I'm bypassing the intermediary layer of typing altogether, which means my hands are free to articulate through gestures and add to the flow of thought.
The body is then differently engaged in the overall dynamics, in comparison to sitting at my computer relatively stiffly.
Gesturing may even affect the thoughts themselves (Goldin-Meadow and Alibali, 2012).
In general, I found it less burdensome to communicate the matters of this thesis through speech than in writing.
Although, I enjoy making technical documentation a lot.

[_____ _] Here

Technologies have a specific intent behind them, they are not arbitrary or neutral, and their prevalence means us having to play on someone else's terms and in accordance with their interests.
This is exacerbated in proportion to the pervasiveness of a specific technological instance and the powers it allocates to the people behind it.
I'm working towards making my own 'blank'/ paranodal (Mejias, 2013) domain, outside the established systems while unavoidably being part of them in many ways, all the while creating a space for publishing, having conversations around diverse autonomous artistic and cultural practices and tools, sharing knowledge and learning together.
As Ulises Ali Mejias outlines in 'Off the Network: Disrupting the Digital World' (2013):

> ---
> ‘the paranodal is not passive; its existence shapes nodes and the relationships between them [much like in urban planning, a "bad" neighborhood "forces" city planners to build a highway around or across it, so that cars can bypass it]. The instability of paranodal space is what animates the network, and to attempt to render this space invisible is to arrive at less, not more, complete explanations of the network as a social reality.’
>
> ---

Parandality in my case entails devising my own autonomous space through which I can communicate, share and interact with others on my own terms.
To me, this started with setting up the servers in my closet, and hosting and modifying several tools on them.
As for the knowledge-sharing part of this undertaking goes, to conduct the online workshops I am using a browser-terminal tool [Terminado, which conjures up a terminal in a browser] and thus opting out of the video conferencing type of sessions.
[My disdain for an enabled camera pointing at me runs deep.]
These workshops are about having a hands-on and proactive approach towards implementation.
Making sure that this is a welcoming place is imperative.

The process of steering away from the imposed environments (Nelson, 2017) brings flexibility while still existing within them.
In this thesis, I am relaying my personal experiences, struggles and joys that come with running the servers at my place and self-hosting several web-based applications and websites.
These efforts are geared towards creating my own space, which is intermittently and in an introverted manner opened up for other people.
Furthermore, I'm reflecting on how these processes shifted my relationship with these tools.
Instability, at times quite painfully, turned out to be an inherent part of my endeavor as you'll learn from this text.
But it is all worth it.
Sometimes I even find these frustrations to be fun.
Sometimes.
Alongside touching on some technical aspects of these processes, I'm delving into the semantics of self-hosting and how it relates to the relevant practices, works and approaches of other individual practitioners and communities to convey many ways of doing servers, networks and opting out of the default modes of engaging with technology / tools.

___._ / Server

I'm running this <<web> <server>> on a <4 gb RAM> <Raspberry Pi> <4 Model B> with <Raspberry Pi OS Lite>.

What is a server?
Probably most people, including myself, have seen the data centers and servers in the movies before knowing what they were.
A clue is in the name -- it serves something.
But before dissecting that part, it's important to make one thing clear: it is always tied to a physical object, even when it is referred to as a 'cloud'.

A Server is an internet-connected computer that provides various kinds of services.
Based on the latter, there are different types of servers: mail servers [for affording email communications], database servers [for hosting and sharing databases], web servers [for hosting websites] and so on.
Depending on the nature of the provided services, the required computing power and security measures, a server can be as small as a single board computer [like a Raspberry Pi, for instance] or they might need a dedicated space or an entire building[s] -- data centers, stocked up with computer systems [for hosting services and storing data].

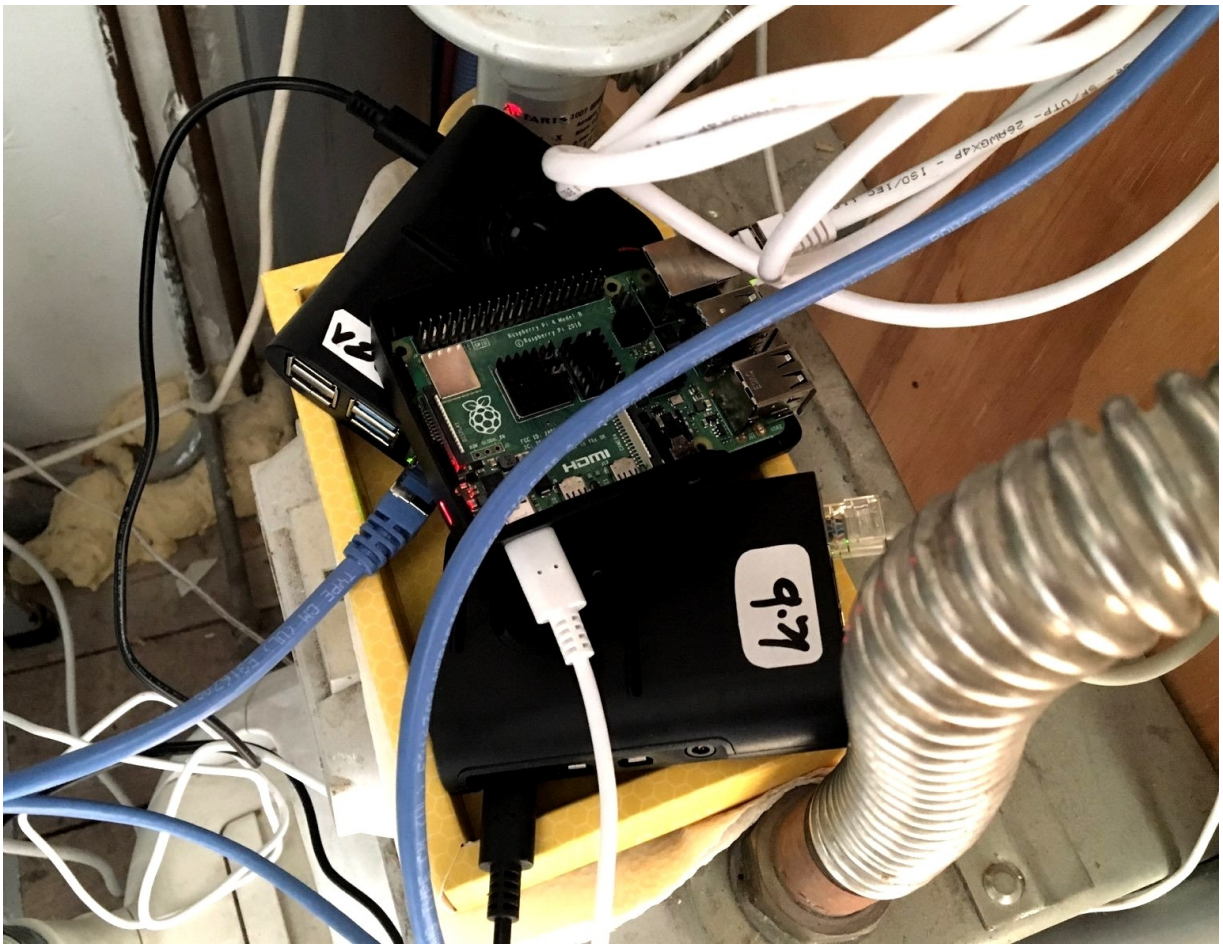
> :// On March 10 a data center building of Europe's largest hosting* [cloud computing and dedicated hosting] providers (OVH, 2021) went aflame, disrupting a few million websites, 'including news sites, banks, webmail services, and online shops selling PPE' (Coble, 2021).
>
> > * OVH specializes in two main types of hosting: cloud computing and dedicated hosting. The former is a server cloud application through with the services are made available. Server cloud refers to the distributed resources, which means that they may be spread over various locations and servers. When it comes to a dedicated hosting, a client is allocated an entire physical server, whereas with cloud hosting you are dealing with a 'virtual' server.
> >
> > But why is it called a 'cloud'?
> > 'Cloud computing' doesn't tell me anything about what is actually going on.
> >
> > > ---
> > > 'The word cloud was used as a metaphor for the Internet and a standardized cloud-like shape was used to denote a network on telephony schematics.' (Cloud computing, 2021)
> > >
> > > ---
> >
> > Of all the computing-related abstractions and metaphors I've encountered so far, to me, 'cloud' and 'cloud computing' must be the most bizarre ones, in that there is no solid hint to the actual underlying processes or a location the 'computing' takes place or the necessary infrastructure that would support it. At least with a 'desktop' or a 'folder' you can somewhat

guess what to expect functionality-wise. One thing 'cloud' conveys accurately though, is its ability to cloud the judgment when you hear about it the first time.
As Robert M Ochshorn aptly puts it in 'notes on intimacy' (2016):

> ---
> 'Abstraction is a technique to help us forget.' [...] 'Consider the hollowness of learning the periodic table without having a deep relationship to its elements.' [...] 'without the concrete, it becomes a smokescreen, a barrier to insight, a dangerous surrogate'
>
> ---

Some abstractions engender forgetting by being deliberately confusing and overwhelming, or by declaring superiority through a sturdy obfuscation. 'Cloud' ticks several of those boxes but it also seems to be an arbitrary metaphor no one bothered to revise, which then stuck around and everyone just has to deal with it.

__ These are my three Raspberry Pi [RPi] servers.



__ This is the OVH data center building I referred to earlier.

To peel off the layers of abstraction, here is some nitty-gritty of one of my servers:

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

$ pinout

,--------------------------------.
| oooooooooooooooooooo J8   +======
| 1ooooooooooooooooooo  PoE |   Net
|  Wi                      oo +======
|  Fi  Pi Model 4B  V1.1 oo      |
|        ,----.               +====
| |D|    |SoC |               |USB3
| |S|    |    |               +====
| |I|    `----'                  |
|                    |C|       +====
|                    |S|       |USB2
| pwr   |HD|   |HD| |I||A|    +====
`-| |---|MI|---|MI|----|V|-------'


Revision           : a03111
SoC                : BCM2711
RAM                : 1024Mb
Storage            : MicroSD
USB ports          : 4 (excluding power)
Ethernet ports     : 1
Wi-fi              : True
Bluetooth          : True
Camera ports (CSI) : 1
Display ports (DSI): 1



J8:
   3V3  (1) (2)  5V
 GPIO2  (3) (4)  5V
 GPIO3  (5) (6)  GND
```

```
 GPIO4  (7) (8)  GPIO14
   GND  (9) (10) GPIO15
GPIO17 (11) (12) GPIO18
GPIO27 (13) (14) GND
GPIO22 (15) (16) GPIO23
   3V3 (17) (18) GPIO24
GPIO10 (19) (20) GND
 GPIO9 (21) (22) GPIO25
GPIO11 (23) (24) GPIO8
   GND (25) (26) GPIO7
 GPIO0 (27) (28) GPIO1
 GPIO5 (29) (30) GND
 GPIO6 (31) (32) GPIO12
GPIO13 (33) (34) GND
GPIO19 (35) (36) GPIO16
GPIO26 (37) (38) GPIO20
   GND (39) (40) GPIO21

For further information, please refer to https://pinout.xyz/

.............................................................
```

pinout neatly displays the circuit board layout and the hardware specifications of this Raspberry Pi. The command output shows the Soc model [system on a chip -- the brain of the device], RAM capacity, RPi model, and so on. You can see that there are 4 USB ports spread over two different locations on the board and that there is one Ethernet port present. The 40-pin GPIO [general-purpose input/output] header is referenced further down the list. Currently, I'm using the 5 volt (4) and the ground pin (6) to power up the cooling fan.

I think I haven't seen a Raspberry Pi until it first caught my eye on a TV series Mr. Robot.
The setting was dramatic: the protagonist, Elliot came up with an idea of connecting an RPi to a thermostat at Steel Mountain [a fictional data center modeled around an actual one -- Iron Mountain (Steel Mountain, 2021)]. This way, they [a hacker group named fsociety] would gain access to the intranet [local computer network] of the facility and raise the temperature in the storage room and subsequently destroy the backups stored on the magnetic tapes [LTO-9 or Linear Tape-Open 9, to be more precise].
For an introduction to a device, this was quite a hectic one:

- __ 'is Raspberry Pi specifically designed to accomplish such tasks?'
- __ 'is it a hacker thing only?'
- __ 'what other kind of 'damage' can it do?'
- __ 'can it run DOOM?'
- __ [...]

It's hard for this little 'menacing' object not to pick a curiosity.
And then you see a bare green circuit board, a chip, the connectors, the pins, the ports -- parts that are usually hidden away under the plastic casings not to 'bother', confuse or inconvenience the casual users.
So, this tiny gadget is a daunting black box, until you find out it's just a small computer.
But this doesn't simplify the practical matters and the initial interaction is still quite awkward.
You may take an easier route by connecting a mouse and a keyboard to it and the RPi to a screen, power it up, and see it boot into its interface.
Another way is a terminal-way, which is how I approached it [because <more terminal> = <more fun>] : this means configuring the RPi via your computer terminal while both machines are connected to the local network.

Last Spring, we worked on the Special Issue XI.
That's when my first proper and up-close RPi encounter happened.
This project involved setting up 3 Raspberry Pi web servers and hosting the static website for the archive we were working on.
This is as far as I can go with the details about the archive itself as the project dealt with the sensitive materials and subject matter.
During one of the sessions, Aymeric [Mansoux] joined us, dropped off three RPi's, and asked us to set them up and prepare for hosting.
He gave us a few notes regarding the minimum requirements, such as going with an NGINX [the web server software which would configure the RPi as a web server], installing minimal Raspbian operating system [because it would suffice for this particular project], removing the default Pi user and disabling a password login [for the security reasons], and going with the SSH keys* instead.

> * To securely and remotely [from your laptop terminal, for inctance] log into a server you can use a cryptographic network protocol called SSH [Secure Shell]. First, you need to generate a public/private key pair [SSH keys] and copy the public key [i.e. the contents of the public key file] onto a remote server [in an authorized_keys file] you want to have access to. So, when you try to log into a server [ssh into it] after configuring it, the keys match and you are in! There are several commands for producing the key pair and here is one of them:
>
> ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
>
> ```
> $ ssh-keygen -t ed25519 -b 320
> ```
>
> ....
>
> To be extra cautious, you may enable a passphrase for the keys [you'll be presented with this option after running the command].
> Now you have two files [id_ed25519 /a private key/ and id_ed25519.pub /a public key/] in a hidden .ssh folder [hidden file/folder names start with a dot] in the home directory.

So, I took one of the RPi's and started figuring it all out.
It took hours at first because it was new territory for me and the online manuals would not quite fit my specific situation at all times:

> Finding RPi IP address on a massive school network after mapping it didn't work out;
> I would get a very long list of IP addresses, but RPi wouldn't show up with its default hostname [raspberrypi] alongside the IP address and neither was I able to ssh into it with the default credentials [user: pi // pwd: raspberry // hostname: raspberrypi]:
>
> ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
>
> ```
> $ ssh pi@raspberrypi
> ```
>
> ....
>
> The next best thing to do was to just connect it to a screen and configure it manually, but that day RPi 'refused to cooperate' because of a subpar HDMI cable.
> I made it work eventually.

I did set up several RPi's a few times and gradually figured out the less complicated ways to approach the process.
This was a dainty way to learn about the networking basics, the IP address allocation, the public/local IPs, ssh, etc., and of course, running an autonomous web server.
I think it is important to note that we knew we could consult with three different tutors in case we got terribly stuck and chances were we might have received several distinct perspectives on a matter at times, which would solidify our overall insight.
The fact that we had this entry point, from which we could engage with the processes and tools, and understand that there is a space to make it your own was a big deal for me.
To me, it snowballed from then on.
I am very much familiar with the scenario where you don't know where to even begin with such practices but feel the strong traction nevertheless.

So, it was invaluable to me to create a space where a multitude of angles and approaches to the implementation of autonomous practices could be shared freely during [and after] the sessions and where the simplest of the questions would be very welcomed.
But the challenge laid in making such space online without me having to compromise or disregard my character: I am introverted, I don't like cameras as I've already mentioned, I am not interested in constant availability, and so on.

I have settled on organizing monthly online sessions inside the terminal in a browser.
The tool that accommodates this particular setup is Terminado [https://github.com/jupyter/terminado].
Terminado emulates an actual server terminal in a browser, and upon running the single.py script in a virtual environment on my server it becomes accessible via a specific URL, meaning if you have the link you have access to my server just like you would if you were able to login into it from your terminal.

This, of course, means opening up the server to an immense vulnerability and it is not only a purely technical [security concerned] one but it is also related to inviting a person[s] onto a private territory, which they will be able to modify and/or damage.
But in case the situation goes haywire, I can terminate the session simply by hitting Ctrl+C and stopping the single.py script.
The reason I wanted to have a 'real' terminal instead of a simulated [front-end-only] one in a browser, was that I wanted to make the infrastructure more tangible: for example, in a session which is about self-hosting a new tool or doing a sysadmin work it is explicit and evident that these operations are impacting this particular server, and at the end of the day you see this tool is run from this server and there are commands to double-check that.
I also leave the terminal / server open for some time after the session is over so that people can travel through the server filesystem to see how it is arranged or to check what had happened here before [by going through the scripts that are already there].
The sessions themselves are split in two distinct directions: workshops and conversations.
Workshops are about configuring autonomous approaches through the tools and methods that support this.
This can be achieved through making a tool, self-hosting and modifying an existing one, maintaining and securing the [server] space through system administration.
The conversations with the invited guests would also cover these matters but understood and told through the personal experiences, discoveries, failures, tips, practices, etc.
There is a dedicated pad [of eth.leverburns.blue] for every session.
Pads are for interaction and documentation.
You can also ask questions, make comments and suggestions, and run commands inside the terminal itself if you prefer to.
To log every new terminal session in a separate .log file and save it in the Shell_logs directory I added one line of code in a .bashrc file:

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||

test "$(ps -ocommand= -p $PPID | awk '{print $1}')" == 'script' || (script -f $HOME/
Shell_logs/$(date +"%d-%b-%y_%H-%M-%S")_shell.log)

..............................................................
```

All the documentation will be available later on at http://terminal.leverburns.blue/

The interview with spideralex I referred to some time ago makes its way back here, as the outlined points are strongly resonant to my approach:

> ---
> 'I think we need to physicalize and visualize servers as a crucial space, once we were saying, we need a room of our own, now we need a ‘connected room’ of our own – that is also a book by Remedios Zafra, a cyberfeminist from Spain that was theorizing and thinking about those dimensions. So, of course, the server is the place where your data is going to be hosted, the contents of your websites, online services you ask for, and then the server serves it to you. But we don't want to be served, we don't want to be always using a provider of a service. I think a feminist server is also a space that we want to inhabit, as inhabitants, where we make a contribution, nurturing a safe space and a place for creativity and experimentation, a place for hacking heteronormativity and patriarchy.'
>
> ---

...__ _ _\\

My Etherpad-hosting server has crashed on several occasions.
The first time it happened was when I ejected an SD card from it, put it in another RPi, and discovered that DHCP* Server refused to assign an IP address to it.

> * DHCP stands for Dynamic Host Configuration Protocol. It is a network management protocol for IP address allocation on a local network. DHCP server uses this protocol to respond to IP address request from a device and assign it a unique IP address dynamically [temporarrily -- DHCP lease]. So, if your laptop IP address today is 192.168.1.20 and you disconnect it from the network for 35 hours, for example, then join the network again, it might not have the same IP address. This depends on the lease time configuration for a particular network. Many household routers, including mine, incorporate DHCP server functionalities, thus I don't need to have a separate dedicated DHCP server.
>
> This command is one of the many ways of finding out your device IP address:
>
> ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
>
> ```
> $ hostname -I
> 192.168.1.4
> ..............................................................
> ```

Devices require an IP address to be able to communicate over a network.
So, In this case, I couldn't ssh into my RPi.
Over the next week, countless attempts went into fixing the situation.
This entailed moving the SD card back and forth between my laptop and RPi while trying out different methods.
I also switched a power supply and a network cable to the different ones to check if either of them were to blame.
I can't even remember how many threads I've pulled, it was no time to pause and document this high-velocity ordeal.
Eventually, I reached out to Aymeric [Mansoux] and Michael [Murtaugh] for help.
They suspected a MAC address* 'confusion'.

> * MAC [Media Access Control] address is a unique hardware identifier hard-coded on the network interface card (NIC) of the device.

In this scenario, the Raspbian OS would have saved the MAC address in the operating system.
Therefore, when I relocated the SD card to a different RPi with its own MAC address and that RPi requested an IP address, DHCP server was left baffled and gave up.
Neither many different iterations of temporary MAC address change, nor static IP address allocation worked.
Chances are, the culprit might have been a faulty, low-quality SD card all along.
After exhausting all my options I decided to reset the server from scratch, losing valuable pads along the way.
This was a quite painful lesson in the importance of doing regular backups and not messing with

unreliable SD cards.
Troubleshooting gives more insight into the inner-workings of the machine and the infrastructure. You are forced to acknowledge and take notice of even the seemingly mundane parts, like cables and then there are aspects and components you had no idea that was there or how they operated.

The reason I wanted to migrate the SD card to a different RPi was that I wanted to spread the computational load on two servers, but mostly for the security objectives:

At a time, I was self-hosting several web pages, Etherpad instance, and Terminado on a single 2 gb RAM Raspberry Pi.
Launching Terminado invites risk as it opens the doors to the server so generously.
Therefore, I decided to have a dedicated the 2 gb RAM server for Terminado and move everything else to a 4g RAM RPi, so that if crud hits the fan it wouldn't take down everything else with it, and besides, there is a limit to how much of the server I am willing to give other humans access to.

To get the the system memory information:

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:           3.7Gi       738Mi       1.2Gi       207Mi       1.9Gi       2.7Gi
Swap:           99Mi       1.0Mi        98Mi
..............................................................
```

Even seeing heavy traffic in auth.log and access.log files made me quite anxious.
I was very suspicious at first, but I got some reassurance that this is nothing out of ordinary.
This is what you get when you connect to the internet: constant traffic and an incommensurable amount of moving parts.

```
||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
```

```
$ sudo tail -f /var/log/nginx/access.log
[sudo] password for ezn:
69.171.251.31 - - [16/Feb/2021:11:33:12 +0000] "HEAD / HTTP/1.1" 200 0 "-"
"facebookexternalhit/1.1 (+http://www.facebook.com/externalhit_uatext.php)"
69.171.251.9 - - [16/Feb/2021:11:38:34 +0000] "HEAD / HTTP/1.1" 200 0 "-"
"facebookexternalhit/1.1 (+http://www.facebook.com/externalhit_uatext.php)"
69.171.251.20 - - [16/Feb/2021:11:38:37 +0000] "HEAD / HTTP/1.1" 200 0 "-"
"facebookexternalhit/1.1 (+http://www.facebook.com/externalhit_uatext.php)"

$ sudo cat /var/log/auth.log
Feb 14 00:00:14 kibkadar sshd[1656]: Received disconnect from 45.145.185.222 port
56840:11: Normal Shutdown, Thank you for playing [preauth]
Feb 14 00:00:14 kibkadar sshd[1656]: Disconnected from authenticating user root
45.145.185.222 port 56840 [preauth]
Feb 14 00:00:34 kibkadar sshd[1658]: Received disconnect from 45.145.185.222 port
34080:11: Normal Shutdown, Thank you for playing [preauth]
Feb 14 00:00:34 kibkadar sshd[1658]: Disconnected from authenticating user root
45.145.185.222 port 34080 [preauth]
..............................................................
```

These are only the excerpts, these lists are very long.
tail -f loops over a file and prints out the updated stream of its contents.
cat outputs the contents of the file or concatenates the files.
access.log file saves the client request [whether it's a bot or a user] data and auth.log file stores the authentication log information.

Maintaining a server / infrastructure involves care and presence, especially since it's expanding and is connected to the internet.
I now have a third RPi just to test things out on before implementing them on any of the other servers.
I have had to reinstall the servers, Etherpad and Terminado one too many times at this point.
This space immediately became personal.
This becomes even more peculiar when hosting the tools for others to use.
But as spideralex points out, 'technology is bound to fail, [it] is going to fail' (Sollfrank, 2018).

A few months ago I had a conversation with dickreckard [their preferred alias] in my browser-terminal.
[Manetta [Berends] introduced us over an email after I asked her if she could put me in touch with LAG [a hacklab in Amsterdam]]
They are working with LAG and I was curious about their operation; it seemed like a welcoming and all-encompassing space where you would be able to 'recycle hardware in the giveawayshop, get help breaking/repairing your devices, install linux on them, taste our mate, discuss the existential threat imposed by ubiquitous connectivity, reclaim your digital life, destroy capitalism, use our machines, make music, make a website, etc.' (LAG, 2019)
With Disroot, in particular, I was wondering what it took to provide the diverse services [email, cloud, chat, pads etc.] with such stability, especially when the infrastructure is community and volunteer maintained.

>>

me: was curious about Disroot and how those tools are hosted
dickreckard: so Disroot is a project coming from people that were involved with LAG and later decided to focus on that project and also the rest of their life, so that did not live so much time to involve in a collective that takes also a lot of time and energies.. so since they wanted to offer services, in a sense, that needed an approach for sure more in line with contemporary computing, like virtual machines dockers stuff like this [...] and that's an example of the different needs/capacities of projects.. to give that type of stable services that you wish to be used by people into movements/political projects around, I imagine that would be difficult within LAG infrastructure for example.. that is more prone to experimentation and sudden issues :D

-- (dickreckard, 2021)

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Complexity and a human aspect of the infrastructure are acutely present in the case of SNET [StreetNet] -- a massive mesh network in Cuba. (Jacobs and Dye, 2020)
It was implemented by the gamer community and eventually grew into a large scale off-grid network.
Sustaining this network is only possible through a persistent social connectivity.

> ---
> 'one limitation of defining the Internet as a large technological system or infrastructure is that this tends to frame the Internet as a channel for transmitting data, rather than as a field of social practice' [...] 'a systems approach also privileges the role of system builders over users'
>
> -- (Abbate in Jacobs and Dye, 2020)
>
> ---

<<»›<<<<< Out

:// Getting to know a terminal and understanding its powers supercharged the thoughts of autonomy in a practical sense.
It's not fancy, it comes with every operating system and it takes you places GUI doesn't want you to see.
But of course, there is a catch: you should know how to 'talk to it'.
Fortunately, learning it remains extremely fun to me.
Through this process, you uncover and remove the masking layers of metaphors the computational realm is so riddled with.

:// And then there are severs and self-hosting.
Having these dedicated physical objects at my place is very comforting.
Especially when I are exposing them to the vulnerabilities [just by connecting to the internet and more so, when launching Terminado].
Knowing that I can just unplug the device in case things go out of hand gives a sense of control over a situation.

:// Opening up this personal space and inviting people to learn together makes me joyous and anxious: there is so much room for the serendipities and errors.
This place is also explicitly and unapologetically introverted; it is open only for a limited time and in a specific interface.
It's ok to not be available 24/7.

:// At the same time, I'm thinking how can this territory be more inhabitable for others, how can it fuel people in a less intense manner than a browser terminal session.
Setting up a Mediawiki instance and a mailing list would be the next steps.

References

Yitbarek, S., 2019. Heroes in a Bash Shell, Command Line Heroes. [podcast] 2019. Available at: < https://www.redhat.com/en/command-line-heroes/season-3/heroes-in-a-bash-shell#> [Accessed 16 March 2021]

Yitbarek, S., 2019. Python's Tale, Command Line Heroes. [podcast] 2019. Available at: <https://www.redhat.com/en/command-line-heroes/season-3/pythons-tale> [Accessed 21 March 2021]

spideralex., 2018. Forms of Ongoingness. Interviewed by Cornelia Sollfrank [video] Creating Commons, 16 September 2018. Available at: <https://vimeo.com/302087898> [Accessed 1 March 2021].

Blevins, A. and Mandeville, Z., 2020. The Map is the Territory. [online] solarpunk.cool. Available at: <https://coolguy.website/map-is-the-territory> [Accessed 14 March 2021].

Drucker, J., 2013. Reading Interface. PMLA, [online] 128(1), pp.213-220. Available at: <https://www.jstor.org/stable/23489280> [Accessed 23 March 2021].

Tech Learning Collective, 2021. Beyond "learning to code": How Tech Learning Collective merges IT training with emancipatory political action. LibrePlanet Conference, 15 March, Online.

Kochuu, 2003. [film] Directed by Jesper Wachtmeister. Sweden: Solaris Filmproduktion

Nelson, T., 1999. Ted Nelson's Computer Paradigm,Expressed as One-Liners. Xanadu, [blog] 29 January. Available at: <https://xanadu.com.au/ted/TN/WRITINGS/TCOMPARADIGM/tedCompOneLiners.html> [Accessed 23 March 2021]

Paul, J., 2020. Your Computer Isn't Yours. sneak.berlin, 12 November. Available at: <https://sneak.berlin/20201112/your-computer-isnt-yours> [Accessed 14 March 2021].

Engst, A., 2020. Apple Network Failure Destroys an Afternoon of Worldwide Mac Productivity. tidbits, 13 November. Available at: <https://tidbits.com/2020/11/13/apple-network-failure-destroys-an-afternoon-of-worldwide-mac-productivity> [Accessed 14 March 2021]

Harmon, E., 2020. RIAA Abuses DMCA to Take Down Popular Tool for Downloading Online Videos. EFF, 5 November. Available at: <https://www.eff.org/deeplinks/2020/11/riaa-abuses-dmca-take-down-popular-tool-downloading-online-video> [Accessed 14 March 2021]

Harmon, E. and Stoltz, M., 2020. GitHub Reinstates youtube-dl After RIAA’s Abuse of the DMCA. EFF, 17 November. Available at: <https://www.eff.org/deeplinks/2020/11/github-reinstates-youtube-dl-after-riaas-abuse-dmca> [Accessed 14 March 2021]

Valsorda, F., 2012. LICENSE. [online] Available at: <https://github.com/ytdl-org/youtube-dl/blob/master/LICENSE> [Accessed 16 March 2021]

Goldin-Meadow, S. and Alibali, M.W., 2012. Gesture's role in speaking, learning, and creating language. Annu Rev Psychol, 25 July. [online] Available at: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3642279/> [Accessed: 22 March 2021]

Nelson, T., 2017. [online] Available at: <https://twitter.com/TheTedNelson/status/928610222544842752> [Accessed: 15 November 2020]

Mejias, U., 2013. Off the Network: Disrupting the Digital World (Electronic Mediations). University of Minnesota Press.

'OVH' (2021). Wikipedia. Available at <https://en.wikipedia.org/wiki/OVH> [Accessed: 22 March 2021]

Coble, S., 2021. OVH Data Center Fire Impacts Cyber-criminals. Infosecurity Magazine, [online] Available at: <https://www.infosecurity-magazine.com/news/ovh-data-center-fire-impacts/> [Accessed: 22 March 2021]

'Cloud computing' (2021). Wikipedia. Available at <https://en.wikipedia.org/wiki/Cloud_computing> [Accessed: 23 March 2021]

Ochshorn, R.M., 2016. notes on intimacy. [online] Available at <https://rmozone.com/snapshots/2016/03/notes%20on%20intimacy.pdf> [Accessed: 23 March 2021]

'Steel Mountain' (2021). Mr. Robot Wiki. Available at <https://mrrobot.fandom.com/wiki/Steel_Mountain> [Accessed: 23 March 2021]

LAG, 2019.]LAG(is a hacklab. [online] Available at: < https://ikiwiki.laglab.org/> [Accessed 25 March 2021].

dickreckard, 2021. [browser-terminal] (Personal communication, 22 February 2021)

Jacobs, A.Z. and Dye, M. (2020). Internet-human infrastructures: Lessons from Havana’s StreetNet. [online] Available at: https://arxiv.org/pdf/2004.12207.pdf. [Accessed: 15 November 2020]