

PEDRO GOMES BRANQUINHO

Free software in industry and academia

Lorena

2021

PEDRO GOMES BRANQUINHO

Free software in industry and academia

This monograph is presented to the Engineer School of Lorena, the University of São Paulo, so to be obtained the title of Bachelor by the Graduation Program on Engineering Physics with emphasis on the Science of Materials.

University of São Paulo - USP
Engineer School of Lorena
Monograph, Conclusion Thesis

Supervisor: Dr. Wei-Liang Qian

Lorena
2021

PEDRO GOMES BRANQUINHO

Free software in industry and academia/ PEDRO GOMES BRANQUINHO. – Lorena, 2021-

Supervisor: Dr. Wei-Liang Qian

Dissertation – University of São Paulo - USP

Engineer School of Lorena

Monograph, Conclusion Thesis , 2021.

1. Free Software. 2. Open Source. 2. Academy and Industry. I. Wei-Liang Qian. II. University EEL-USP. III. Escola de Engenharia de Lorena. IV. Free Software on Industry and Academia.

*To those whom found me in their own path
and, by finding me, made part of my own.*

Acknowledgements

My acknowledgments wouldn't fit in a single page. But, for the purpose of conciseness, I will mention those who are closer to my work. I thank first my advisor, Wei-Liang Qian, who in his patience and kindness knew how to conduce me to produce the present work. I thank Juan Zapata for the support and enthusiasm on teaching Mathematics and showing me the way of how to study it myself. Last but not least, I thank Luiz Eleno who has been a role-model for me, and has teach me so much about computing through out the years.

And, in a big umbrella, I thank all those anonymous people who have contributed for my experience of communal sharing and understanding in the Open Source community. Specially, David Wilson, the founder of System's Crafter, from whom I derived the basis of my Emacs's system.

Abstract

In this work, It's shown how to build a series of application only upon a Free Software system - EXWM, Artix Linux OSS. I explain how the experience of participating in the Open Community can be significant for other Engineers; specially Physics Engineers. It's delineated what are the current trends on the adoption of Free and/or Open Source Software (FOSS). Furthermore, I put forward some of the tools I used in Academia, and in Industry, and some other not so well known software, which could be used in these two contexts - e.g., Freqtrade, OR-Tools, Git(hub) et cetera. I also argue why Linux is such a key software for someone shifting to the Open Source paradigm.

Key-words: trends. foss. academia. industry. linux. freqtrade. or-tools. git. github.

Resumo

Demonstrou-se como é possível construir uma série de aplicações baseada em softwares de licença livre, à partir de um sistema aberto, o Linux com interface EXWM - Emacs X Window Manager. Além disso, foi propiciado casos reais de aplicações na Indústria e no investimento privado, autônomo. Bem como, utilizações na Academia, à nível de lecionar, e pesquisa. Sustenta-se que a economia aberta possui similaridade estrutural ao movimento *Open Source* e seu desenvolvimento, o que aponta que essa é e continuará a ser, paulatinamente mais, o paradigma de desenvolvimento econômico tecnológico. Assim, imprescindível à formação do engenheiro.

Palavras-chaves: software livre. automação. freqtrade. indústria. academia.

List of Figures

Figure 1 – Schema of a tower of interpreters	12
Figure 2 – Categorization of the study of towers of interpreters	12
Figure 3 – Torre de interpretadores e os Sistemas Operacionais.	13
Figure 4 – Genealogy of Linux's Distributions	15

List of abbreviations and acronyms

FOSS	Free and Open Source Software
abnTeX	ABsurdas Normas para TeX

Contents

1	INTRODUCTION	10
1.1	Objective	10
11	section.1.2	
1.2.1	Why does GNU/Linux matter?	11
1.2.2	How high level applications benefit from an OS	11
2	BIBLIOGRAPHY REVIEW	14
2.1	Open Source	14
2.1.1	Diversity	14
2.2	GNU/Linux	15
	BIBLIOGRAPHY	16

1 Introduction

In training a Physics Engineer, of which, by definition, is a generalist professional. The use of Free and Open Sourced Softwares (FOSS), as well as the participation in the Open Source community (OS), are two detrimental experiences to have.

The variability, which open source software (OSS) brings to existence of applications and it's extension, can change altogether user's experience. Thus, taking him closer to acting as a developer. This experience of interloping user and developer roles doesn't require that you are a computer scientist or a Information Technology (IT) professional by training. For, programming can be seen as both a Science and an Art (KNUTH, 1968) - e.g., an exercise of self-expression.

OSS guarantees four fundamental liberties [section 2.1](#), the right to study, copy, modify and redistribute it.

Just as the scientific enterprise benefits, with it's rapid development, by means of the global community's participation, which holds space for individuals with a variety of different training. Also, the computation enterprise benefits from the variety of people's training, which constitute the body of the open source community.

1.1 Objective

We will demonstrate the vality of the hypothesis that a engineer professional, without strong training in computation, lags behind it's current potential. Furthermore, we present a range of different applications developed to the state of the art, which are distributed under open licenses (copy-left). Thus, reinforcing the uniqueness of open source phenomena and the need for it's use - the logic and dynamic FOSS brings has no parallel on the economy or scientific community (HIPPEL; KROGH, 2003; PETERS, 2009).

There exits debate around the meaning of *Open Science*. This term recently became popular and which has a obvious reference to the *Open Source*. Although, in the social literature, it's a phenomena poorly depicted and rarely debated through the point view that the Open Source Movement has anything to do with it. We cite the most cited article on Google Scholar research on the topic "Open Science" - called "The future(s) of open science" - and which only uses trice the term "Open Source" and in dismissive way (MIROWSKI, 2018).

Therefore, I argue that a strong basic knowledge, for engineer training, is fundamental for understanding the movement and how it has been shaped, so to critically asses the validity of the current social-economic shift we live in. This understand can, in turn, shape

the carer path and formation of the working engineer.

1.2 The interconnection between Applications¹ and the OS

1.2.1 Why does GNU/Linux matter?

We will discuss, as a brief introduction, what is the Operational System (OS) GNU/Linux, and why it's the *de facto* opening door to the Open Source Community. Firstly, the GNU/Linux is the first and most successful project carried out in the Open Source paradigm (TU et al., 2000; WEST; DEDRICK, 2001). Therefore, it's use is a way to acquaintance, in practical terms, with how a business dependent on mainly using the open sourced development products might operate (FINK, 2003).

Furthermore, the new user-developer of the GNU/Linux framework must inherently learn about the accompanying software which comes with it's distribution - which increases it's chance of adoption (WEST; DEDRICK, 2001). This way, the user gradually becomes accustomed to participate on development and extend programs (HERTEL; NIEDNER; HERRMANN, 2003).

Beyond it's initial philosophical appeal to user liberty, GNU/Linux is today's standard in Technological Enterprises. And, although fundamentally opposed of many age-old *modus operandus* of the *status quo* of companies, there is no doubt left under analysis of the benefits it brings to the companies, general economy, and the society triad (MOODY, 2009). To make use of GNU/Linux, thus, is a way to be inserted in this new economic paradigm (HIPPEL; KROGH, 2003; PETERS, 2009).

1.2.2 How high level applications benefit from an OS

In the hierarchy of software and applications, the Operational Systems (OSes) can be seen as a meta-application or meta-software.

"The evaluator, which determines the meaning of expressions in a programming language, is just another program." (ABELSON; SUSSMAN, 1996)

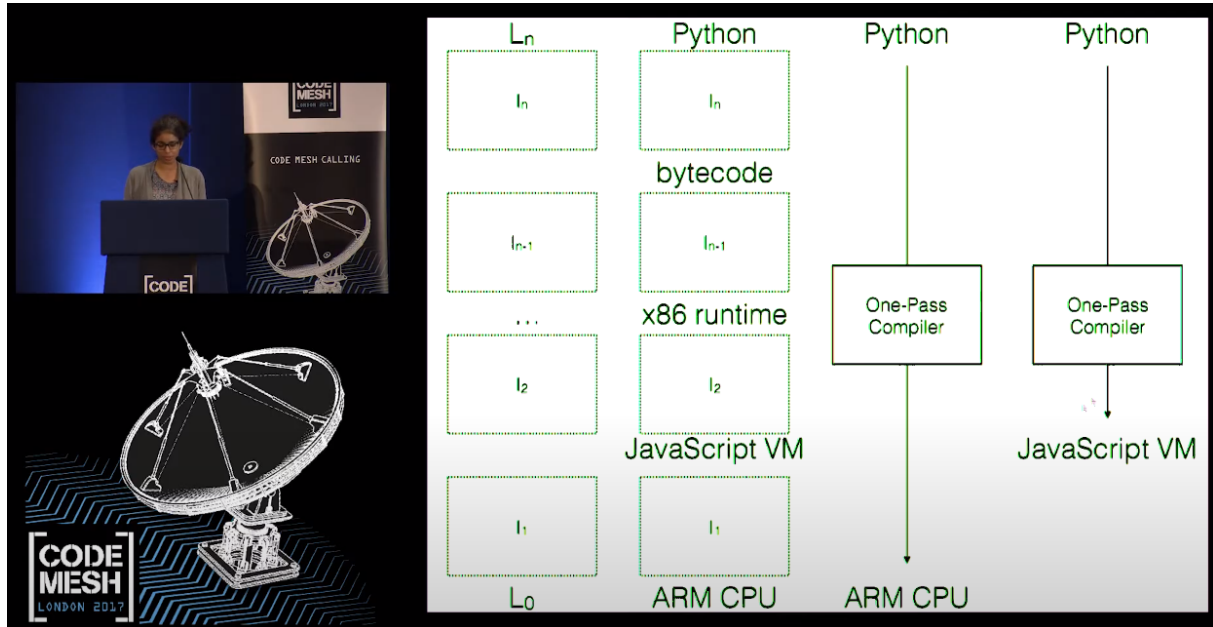
There exists levels, or layers, of abstractions in virtually any application. That is, the concept of meta-programming and Towers of Interpreters comprise a common situation, for which a devoted field of study exists. Thus this area has direct implication for software development practice, as it's a ubiquitous problem faced in computing.

Any OS, as the GNU/Linux, comprise an essential layer in this tower of interpreters. Particularly, an OS communicates with *firmwares* - low-expressivity and highly-performing software, which control *hardware*. Also, they communicate with high-expressivity

¹ An Application comprises the end product of software development.

software, among which contain the user-developer written or extended software. Therefore, the OS play a fundamental role, mediating between low and high level software. This function categorizes them as a *middleware* software.

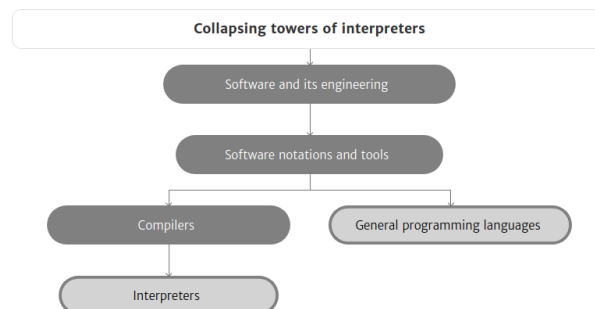
Figure 1 – Schema of a tower of interpreters



Code Mesh, presentation “Towers of Interpreters”, by Nada Amin

The characteristic problem of concatenate a system of software, one on top of the other, introduces complexity to maintaining compatibility among program’s versions and it’s performance. The study of these behavior and it’s theoretical solutions posses a field of it’s own. And, this field is autonomous, detached, for an example, from which languages compose the Tower of Interpreters; or which type of application we are dealing with (AMIN; ROMPF, 2017). The object of study is the final behavior of the system, and if it’s a collapsible system.

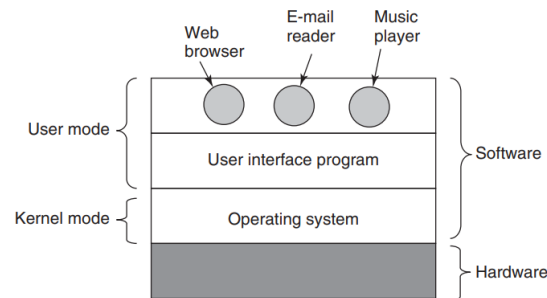
Figure 2 – Categorization of the study of towers of interpreters



Reference: (AMIN; ROMPF, 2017)

Finally, the OSes consist in a big tower-collapse of interpreters. They are subordinate to collapsing firmware, middleware and high-level software. Therefore, as well as the OS conduct this task, as much the user-developer experience is facilitated.

Figure 3 – Towers of interpreters and the Operational Systems.



Reference: ([TANENBAUM; BOS, 2015](#))

If every application programmer had to understand how all these things work in detail, no code would ever get written. Furthermore, managing all these components and using them optimally is an exceedingly challenging job. For this reason, computers are equipped with a layer of software called the operating system, whose job is to provide user programs with a better, simpler, cleaner, model of the computer and to handle managing all the resources just mentioned. ([TANENBAUM; BOS, 2015](#))

2 Bibliography review

2.1 Open Source

Any program which permits the user-developer to have the following liberties:

1. The right to run the program, as seen fit, for any end.
2. The right to access the source code and study it.
3. The right to copy and redistribute it.
4. The right to modify the software.

Practically, the Open Source community fundamentally base itself upon the free distribution of it's tools and programs. One of the differential advantage of having innumerable other people extending the same software is that the advancement of the frontier of the program, in many directions, increases rapidly in relation to a program controlled by a limited number of programmers.

2.1.1 Diversity

Given that one fundamental right of OSS is the modification and propagation of new modified versions. This right implies in the observable wide range of maintained versions of these software, which doesn't have a parallel in any other technological enterprise.

For an example, one key application in any user's computer is a general Graphical User Interface (GUI)'s manager, commonly known as Window Manager (WM). These can be both Floating or Tilling, or mixed WM, e.g., Floating WM are those that the user must hover windows and adjust them manually; Tilling WM are those that a pre-defined program have a set of rules to resize automatically the windows in a screen.

While private Operational Systems (OS), as Windows and MacOS, have frequent releases - a total of twenty five releases for Windows. Generally, they've few *active* versions; Windows have currently four ([MICROSOFT...](#), 2021). MacOS also have four active versions ([MACOS...](#), 2021).

The fact there are only narrowly supported versions is due to, among many contributing factors, users lack the right to alter and extend the software's behavior. Therefore, they fall victims of discontinued support and restrictive access to the company's official upgrades.

On the other hand, there exists, in parallel, around two hundred seventy eight available distributions of Linux ([LINUX...](#), 2021). Of which, there are main/root distributions, which each embody a set of different principles; theoretical and practical philosophies of

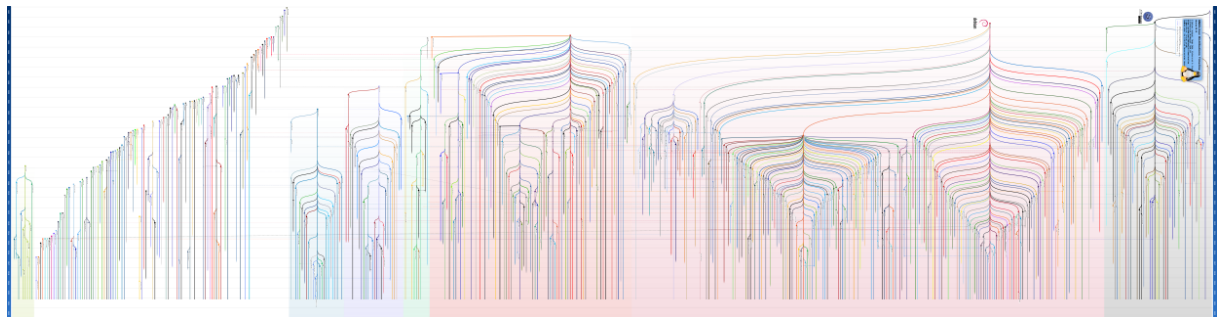
how to extend software.

Thus, just as with any other scope of software, the variability of FOSS always will be grater than monopolized ones.

2.2 GNU/Linux

There are root distributions of Linux, from which many other distributions emanate. Generically, these partitions are called families. We cite some of the most influential and popular ones, Red Hat Linux (☞), Debian (©), CentOS (⚙), Fedora(Ⓜ), Pacman-based (Ⓐ/Ⓜ), OpenSUSE (☞), Gentoo-based (☞), Ubuntu-based(©), Slackware (Ⓢ), Open Sourced-based and the Independent Distributions (☞/Ⓐ).

Figure 4 – Genealogy of Linux’s Distributions



Genealogical history of Linux Distributions ([LINUX...](#), 2021)

Bibliography

ABELSON, H.; SUSSMAN, G. J. *Structure and interpretation of computer programs*. [S.l.]: The MIT Press, 1996. Citado na página 11.

AMIN, N.; ROMPF, T. Collapsing towers of interpreters. *Proc. ACM Program. Lang.*, Association for Computing Machinery, New York, NY, USA, v. 2, n. POPL, dez. 2017. Disponível em: <<https://doi.org/10.1145/3158140>>. Citado na página 12.

FINK, M. *The business and economics of Linux and open source*. [S.l.]: Prentice Hall Professional, 2003. Citado na página 11.

HERTEL, G.; NIEDNER, S.; HERRMANN, S. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. *Research policy*, Elsevier, v. 32, n. 7, p. 1159–1177, 2003. Citado na página 11.

HIPPEL, E. v.; KROGH, G. v. Open source software and the “private-collective” innovation model: Issues for organization science. *Organization science*, Informs, v. 14, n. 2, p. 209–223, 2003. Citado 2 vezes nas páginas 10 and 11.

KNUTH, D. E. The art of computer programming, vol 1: Fundamental. *algorithms*, p. 187, 1968. Citado na página 10.

LINUX, list of distributions. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/List_of_Linux_distributions>. Citado 2 vezes nas páginas 14 and 15.

MACOS version history. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/MacOS_version_history>. Citado na página 14.

MICROSOFT, list of operating systems. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/List_of_Microsoft_operating_systems>. Citado na página 14.

MIROWSKI, P. The future (s) of open science. *Social studies of science*, SAGE Publications Sage UK: London, England, v. 48, n. 2, p. 171–203, 2018. Citado na página 10.

MOODY, G. *Rebel code: Linux and the open source revolution*. [S.l.]: Hachette UK, 2009. Citado na página 11.

PETERS, M. A. Open education and the open science economy. *Yearbook of the National Society for the Study of Education*, v. 108, n. 2, p. 203–225, 2009. Citado 2 vezes nas páginas 10 and 11.

TANENBAUM, A. S.; BOS, H. *Modern operating systems*. [S.l.]: Pearson, 2015. Citado na página 13.

TU, Q. et al. Evolution in open source software: A case study. In: IEEE. *Proceedings 2000 International Conference on Software Maintenance*. [S.l.], 2000. p. 131–142. Citado na página 11.

WEST, J.; DEDRICK, J. Open source standardization: the rise of linux in the network era. *Knowledge, Technology & Policy*, Springer, v. 14, n. 2, p. 88–112, 2001. Citado na página [11](#).