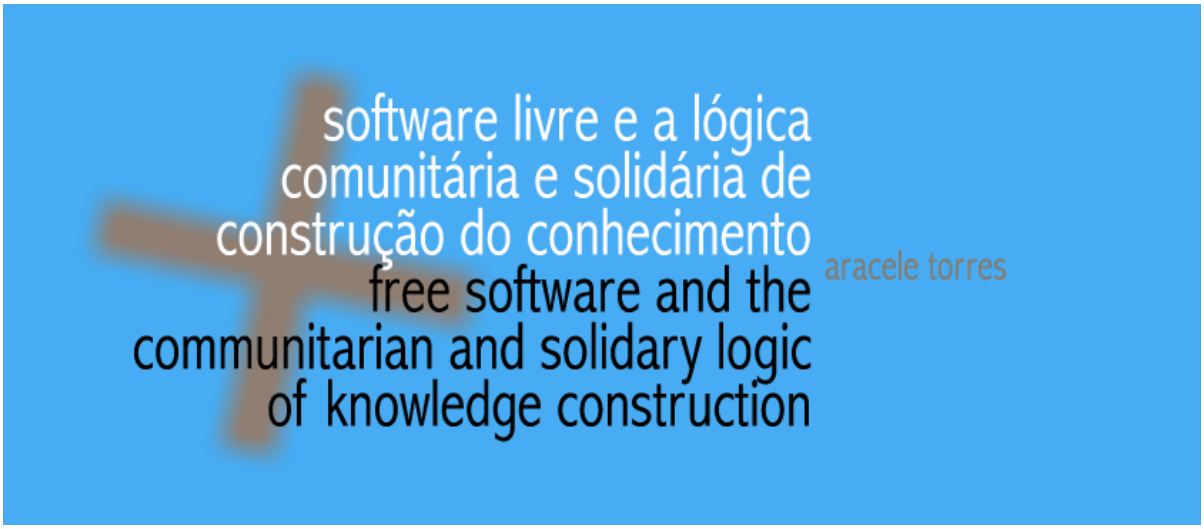
V!18
issn 2175-974x | ano 2019 year
semestre 01 semester



Aracele Torres é historiadora e Doutora em História Social. Atua nas áreas de História da Ciência e da Tecnologia desenvolvendo trabalhos sobre história da tecnologia digital, software livre, ciber-libertarianismo, ideologia e utopia.

TORRES, A. L. Software livre e a lógica comunitária e solidária de construção do conhecimento. **V!RUS**, São Carlos, n. 18, 2019. [online] Disponível em: http://www.nomads.usp.br/virus/virus18/?sec=5&item=98&lang=pt/p>

Resumo

Esse artigo aborda, de uma forma breve, como a noção de compartilhamento e criação coletiva presente no modelo de produção do software livre está vinculada a uma ideia de direito e responsabilidade social. Demonstramos como através da criação do Projeto GNU, que deu início ao movimento software livre, Richard Stallman, procurou se opor a uma lógica anti-comunitária e anti-solidária imposta pela indústria de software a partir do final do século passado. Essa lógica ao transformar o software em um bem privado, de uso e circulação restritos pelo *copyright*, privilegiou a ação, a criação e o ganho individual em detrimento do coletivo. Na contramão dessa tendência, o movimento software livre apresenta uma defesa não somente do conhecimento livre como um valor importante na sociedade, mas também da sua produção de forma colaborativa como essencial para que ele, e a sociedade como um todo, prospere.

Palavras-chave Software Livre, Obra Coletiva, Colaboração, Responsabilidade Social

ARTIGO DE AUTOR CONVIDADO

1 Introdução

O modelo de produção aberto e colaborativo do Projeto GNU (GNU's Not Unix!)[1], que deu início ao movimento software livre, é um marco na história da computação moderna e representa também uma rachadura na lógica de produção individualista neoliberal. Esse modelo propõe que os programas de computador mantenham seus códigos-fonte[2] abertos a qualquer um que deseje: executá-los para qualquer propósito; distribuir cópias desses programas a quem quer que seja; estudá-los para entender o seu funcionamento e poder alterá-los; e redistribuir as cópias de versões modificadas desses programas. Essas são as famosas quatro liberdades básicas de um software livre, que estão relacionadas com as suas permissões de uso e não com o seu preço. Um software livre não necessariamente é gratuito ou não comercial[3]. Essas liberdades não somente permitem como forçam legalmente, através do uso da licença GPL

(General Public License)[4], que os softwares sejam tratados como uma criação coletiva, que está em constante processo de aprimoramento e que permite a participação e colaboração de todos.

A licença GPL, também apelidada de *copyleft*[5], obriga quem pretende redistribuir as modificações feitas em um software livre a também usar a mesma licença livre adotada pelo autor original do software. Isso garante que ninguém possa pegar o código aberto de um software livre, fazer alterações nele e redistribuir o programa com esse código fechado. Essa é uma característica da GPL que um dos executivos da Microsoft chamou certa vez, em tom pejorativo, de "viral", mas que é, na verdade, o segredo de sucesso dos projetos de software livre. A GPL usa o próprio mecanismo do *copyright*, ao qual ela se opõe fortemente, para burlá-lo e inverter a sua lógica, fazendo com que ele ao invés de restringir a cópia e redistribuição funcione como uma restrição a essa restrição (TORRES, 2018, p. 151). Se eu escrevo um software e o licencio sob a GPL, que basicamente é um *copyright*, eu estou dizendo ao mundo que eu gostaria que todos pudessem ter acesso ao código do meu software, e que caso alguém decidisse alterá-lo e redistribuí-lo essa pessoa também seria obrigada a fazer o mesmo.

O fato de o código do programa ser aberto e permitir constantes modificações e melhorias faz do software livre uma obra de construção coletiva. Uma das principais consequências disso foi o desenvolvimento de comunidades em torno de projetos específicos de software livre, como é o caso das comunidades KDE, Debian, Firefox e VLC. A dinâmica de contribuição nessas comunidades varia de acordo com diferentes fatores como, por exemplo, as diretrizes de governança de cada uma delas, o seu tamanho, as orientações político-filosóficas de seus membros. É muito comum que as pessoas envolvidas nelas, os chamados colaboradores ou contribuidores, sejam voluntários e dediquem seu tempo a contribuir para esses projetos sem exigir nenhuma contrapartida financeira. Mas também é comum que empresas ou instituições sem fins lucrativos, como é o caso da Fundação Mozilla, responsável pelo navegador Firefox, remunere pessoas para trabalhar na manutenção do projeto. E embora isso aconteça, existem ainda milhares de contribuidores espalhados pelo mundo que colaboram de forma voluntária no desenvolvimento e manutenção do Firefox.

Esse modelo de produção e distribuição dos softwares livres, no entanto, é o oposto do praticado pela indústria hoje, que adotou, pelo menos desde os anos 1980, como seu padrão o software proprietário ou de código fechado. Neste modelo, a empresa ou indivíduo produz um software e usa mecanismos legais de propriedade intelectual, em geral o *copyright*, para tornar o seu código-fonte fechado, ou seja, inacessível a qualquer usuário que deseje entender como aquele programa foi construído e funciona. Neste caso, só o detentor do *copyright* tem o direito legal de acessar esse código e alterá-lo, assim como também de distribuir cópias desses programas. Caso o usuário tente fazer qualquer uma dessas coisas, sem a devida permissão, ele é enquadrado como violador de um sistema de propriedade intelectual, uma prática que é criminalizada e pejorativamente chamada de "pirataria" pela indústria.

O que prevaleceu como padrão da indústria foi, portanto, uma abordagem do software como uma obra ou bem privado com um fim essencialmente comercial, um produto que precisa ter sua circulação e uso controlados, já que isso impacta diretamente no lucro que os seus donos podem ter sobre ele. Em oposição a isso está a visão do software livre de que um programa de computador deve ser visto como um conhecimento, portanto, uma obra de uso e construção coletiva, um bem público ou comum (commons), que deve ter, antes de tudo, fins sociais. Isso não quer dizer, no entanto, que ela não possa ser comercializada, mas que antes de tudo ela tem uma função social que deve beneficiar a todos e não apenas a uma elite econômica. Apesar de ter se tornado heterogêneo[6] ao longo do tempo, o movimento em defesa do software livre nasceu como um movimento social em defesa do conhecimento livre na sociedade.

2 O surgimento do software proprietário

O conceito moderno de bem comum (*commons*) foi introduzido na literatura acadêmica em 1968, pelo ecologista Garrett Hardin, em seu famoso artigo "The Tragedy of Commons", que abordava a gestão de um recurso finito de uso comum por indivíduos agindo de acordo com seus próprios interesses. No sentido empregado por ele, "commons" refere-se a recursos naturais compartilhados, como rios e terras, que na sua opinião sempre tendem a se esgotar por falta de um uso racional pelos indivíduos. O emprego original desse termo remete ao contexto da Grã-Bretanha medieval e foi utilizado pela lei britânica para descrever as terras de uso comum para camponeses e nobres. Na década de 1990, o conceito foi apropriado pela economia, especialmente após a publicação de "Governing the Commons: The Evolution of Institutions for Collective Action", de Elinor Ostrom, que se opunha a tese de Hardin de que os "commons" sempre tendiam ao fracasso. Atualmente, o conceito tem sido amplamente usado para se referir a recursos culturais digitais (bens imateriais ou intangíveis) que são produzidos e usados de forma coletiva, como a Wikipedia ou o software livre.[7]

Os bens públicos, definidos pela teoria econômica neoclássica, são bens não disputáveis, também chamados bens não-rivais. Não-rivais, porque o seu uso por alguém não impede o uso por outros. Um bem público

também tem a característica de não-exclusividade, ou seja, não é possível impedir que alguém o consuma. Em outras palavras, eles podem ser obtidos sem a necessidade de um pagamento direto pelo seu uso. Em oposição a isso, bens rivais e exclusivos são considerados bens privados. Entretanto, a não-exclusividade não é um atributo inerente aos bens públicos, é um atributo de caráter político, relacionado às instituições que regulam seu uso. De modo geral, quase não há bens que não possam ser privatizados por meio de decisões legais e políticas. De fato, esse processo de privatização do conhecimento, por exemplo, faz parte da lógica de extração de valor do capitalismo contemporâneo (PRADO, 2005).

A adoção do software proprietário como padrão da indústria foi, portanto, uma decisão legal e política, medida tomada em um contexto específico de reestruturação capitalista e avanço do neoliberalismo nos anos 1980. Essa reestruturação levou a um processo de comoditização de bens intangíveis, como a ciência e a tecnologia, que foram transformados, de forma artificial, em bens rivais e exclusivos. A saída para a crise estrutural capitalista do fim do século passado, que culminou em uma queda no lucro e produtividade, foi a criação de novas formas de acumulação de capital. A tecnologia e a ciência foram usadas tanto para aumentar a eficiência do processo produtivo (e assim o lucro), quanto como um produto desse próprio processo (DUMÉNIL; LÉVY, 2003). O capitalismo se revitalizou ao garantir que esses bens comuns fossem transformados em propriedade privada (HARVEY, 2003). Semelhante ao que foi descrito por Karl Marx (2013) no processo de acumulação primitiva do capital, ocorrido na Grã-Bretanha do século XV, onde as terras de uso comum foram transformadas em propriedade privada da nobreza; um movimento de fechamento de bens de uso comum ocorreu no fim de século passado.

Esse fenômeno, chamado por David Harvey (2003) de "acumulação por despossessão", trata-se da transformação de bens culturais, históricos e intelectuais em mercadorias. Através desse processo, empresas produzem e vendem esses bens principalmente em um modelo de aluguel, onde o comprador paga pelo direito de uso desse produto, mas não se torna exatamente o seu dono, ou seja, não pode fazer o que bem desejar com ele. As empresas, portanto, exercem um monopólio sobre esses bens, como é o caso do software de código fechado. E esse monopólio é garantido graças às leis de propriedade intelectual, criadas com a justificativa de "proteger" os direitos do autor e "encorajar" a criação e disseminação das obras, mas que na verdade têm a função de criar uma "escassez artificial" desses bens.

Quando Richard Stallman, o programador estadunidense que iniciou o movimento pelo software livre, anunciou em 1983 a sua ideia de criar um sistema operacional completamente livre, o GNU, construído de forma aberta e colaborativa, ele não necessariamente inaugurava uma tendência. Isso porque era comum até os anos 1970 compartilhar o código dos softwares nos ambientes das universidades e das empresas. No entanto, ele iniciava uma disputa que garantiria um lugar importante na indústria da computação para esse modelo de produção[8]. Assim como fomentava o desenvolvimento de toda uma cultura do "copyleft" que vai além da computação e hoje é adotada por outras áreas como a música, as artes, a ciência, etc (KELTY, 2008). Em seu famoso e-mail anunciando o Projeto GNU, ele usa como argumento fundamental para sua empreitada a noção de que fazer isso era um dever moral para com a comunidade de programadores da qual fazia parte e para com a própria sociedade como um todo. Ele estava assumindo o compromisso de não deixar essa cultura da colaboração, "tão velha quanto os computadores" como ele próprio disse (2002, p. 17), morrer e ser substituída pelo padrão proprietário e fechado que temos hoje na indústria.

Até os anos 1970, o compartilhamento do código dos programas era uma prática comum tanto no ambiente das empresas quanto no das universidades. A computação nesse momento ainda era muito centrada no hardware e a indústria de software era ainda incipiente. Não havia um consenso entre as empresas de que se poderia lucrar com o software tanto quanto era possível com o hardware. Dos anos 1970 para os 1980 o cenário, no entanto, se altera, com, entre outras coisas, a criação dos computadores pessoais; a criação dos primeiros cursos de Ciência da Computação e a consequente oferta de mão de obra especializada; o desenvolvimento de linguagens de programação; o avanço de um discurso neoliberal defendendo expansão do mercado. De modo que as empresas começam a enxergar no software uma possibilidade de lucro e repensar a abordagem em torno dele, passando a adotar *copyright* para restringir a sua circulação e uso e assim ser capaz de lucrar com sua venda. Antes elas geralmente forneciam o software junto com o hardware, sem nenhum custo adicional, e não viam muito problema em permitir que os usuários alterassem os códigos desses programas e os compartilhassem (TORRES, 2018).

Foi contra esse cenário de fechamento dos softwares que o projeto de software livre criado por Richard Stallman se opôs. Stallman desejava manter o software não só como um bem de uso comum, mas também de produção coletiva. A lógica era estabelecer uma espécie de ecossistema de colaboração, onde a ideia de somente usufruir do trabalho de outra(s) pessoa(s) desse lugar à de trabalhar junto com esse(s) autor(es). Um ecossistema onde é possível ser parte também da criação dessa obra que é o software e ter uma responsabilidade social na sua manutenção como uma obra aberta e coletiva, pertencente a todos. Essa é uma noção que vai de encontro ao que prescreve o neoliberalismo, que prega um anti-comunitarismo na medida em que incentiva o indivíduo a trabalhar de forma isolada da comunidade, e defende o discurso de

que o indivíduo se faz sozinho (self-made man), vence sozinho na sociedade e não necessariamente em comunhão e cooperação com os outros, mas na concorrência e na competição.

No contexto do software livre, o autor tem uma responsabilidade social para com a comunidade na qual está inserido, compartilhar o conhecimento é uma "regra de ouro", como colocou Stallman ao anunciar seu projeto (2002, p. 31). No modelo neoliberal a responsabilidade do autor é para consigo, não devendo necessariamente nada à comunidade na qual está inserido, já que o que prevalece é a ideia da meritocracia. Se ele adquiriu certo conhecimento, um saber-fazer, foi através de seu esforço individual e ele tem todo o direito de monopolizar o acesso a esse conhecimento e tirar proveito disso, já que no discurso neoliberal a propriedade privada é direito natural do indivíduo. Não há uma perspectiva de "solidariedade social" ou um "espírito de comunidade", termos muito usados por Stallman na justificativa da importância do seu projeto.

3 Compartilhar como um direito e uma responsabilidade social

Richard Stallman argumenta que temos o direito de fazer uso das tecnologias digitais em formas socialmente úteis, aproveitando os benefícios oferecidos pela facilidade de cópia de arquivos digitais. O compartilhamento de informações digitais é muito diferente do compartilhamento de coisas materiais. Enquanto bens imateriais, como um livro digital ou um código de programa, são bens não-rivais, bens materiais são o oposto. Se alguém tiver um livro impresso e quiser emprestá-lo para um amigo, apenas um deles pode ter este livro naquele momento. Por outro lado, se este livro é digital ambos podem tê-lo e lê-lo ao mesmo tempo. Uma vez que o software é um conhecimento, Stallman acredita que este deve estar disponível para uso por todos. Se alguém possui um programa de computador e quer compartilhá-lo, quem deve ter o direito de decidir se esse compartilhamento pode ser feito ou não: os indivíduos envolvidos ou os detentores do *copyright* do programa? Para Stallman, na sociedade atual essa resposta é baseada no critério econômico de maximização do lucro do proprietário do software, geralmente uma empresa. Quando, de fato, deveria ser baseado em um outro critério, a prosperidade que é alcançada através da partilha de conhecimento. Em sua opinião, quando você é impedido de compartilhar conhecimento, toda a sociedade é prejudicada.

Embora as tecnologias digitais facilitem a reprodução da informação, os direitos autorais ou qualquer outra forma de restrição imposta pelo sistema de propriedade intelectual, no entanto, movimenta-se na direção oposta, bloqueando essa facilidade. O *copyright* funciona a partir da ideia de que existe um proprietário, que detém o monopólio sobre a cópia e distribuição de uma obra e é quem decide legalmente o uso que pode ser feito dela. Os interesses desse proprietário nem sempre coincidem com os interesses da sociedade em geral. O ato de restringir o acesso a uma obra por meio de direitos autorais, reflete a intenção de produzir uma escassez artificial dessa obra. Há um conflito de interesses, uma tensão entre um direito social, o livre acesso ao conhecimento, que Stallman acredita que temos, e o privilégio de um monopolista de restringir esse acesso por lucro.

A distinção entre o detentor do *copyright* e autor do código é muito importante aqui porque esclarece sobre a relação de trabalho e de poder dentro dessa estrutura de produção do código proprietário. O autor do código de um software, no geral, é o trabalhador (no caso a pessoa que programa) que o escreve e não a empresa ou empresário que o vende e lucra com isso. Há no processo de produção do software proprietário a apropriação de um trabalho alheio. O programador ou programadora detém o conhecimento para produzir um software, mas não os direitos de uso ou reprodução desse software, que é alienado ao patrão que os contratou. No caso da produção de um software livre, essa lógica pode ser rompida, já que o autor de um código que é livre continua detendo os direitos de uso e distribuição desse software que ele escreveu, a obra depois de feita continuará sendo dele, mas ele expande esses direitos a todos da sociedade, transformando o software em um bem comum do qual todos podem também usufruir.

Richard Stallman acredita que os interesses da sociedade como um todo devem estar acima dos interesses particulares e que muita importância é dada ao autor na sociedade contemporânea. De acordo com sua visão, essa importância resulta em uma perda no desenvolvimento da produção coletiva e colaborativa do conhecimento, os direitos do autor são colocados acima dos direitos de toda a sociedade. Para salientar que essa importância é produto do nosso tempo e do sistema econômico em que vivemos, ele reforça a ideia de que o fator mais importante em uma sociedade é a produção social do conhecimento, traduzida da ideia de que uma obra deve servir primeiramente a propósitos sociais, em vez de propósitos individuais de quem a criou.

Stallman explica que ao lado do argumento de que o autor tem um direito natural sobre seu trabalho, um direito que iria além do direito social de todos de acesso ao conhecimento; a indústria de software também usa um argumento econômico para justificar seu monopólio sobre o código-fonte. Este argumento pode ser resumido a partir da ideia de que, se não produzirmos software proprietário, os softwares acabarão, porque os desenvolvedores não trabalhariam sem a garantia de que seriam recompensados. O erro desse discurso, segundo ele, é assumir que não existem outros sistemas alternativos ao software proprietário. Além disso, também é um grande equívoco relacionar a existência de software, ou qualquer conhecimento produzido pela

sociedade, com a ideia de propriedade. Software e propriedade privada não são sinônimos, como ele destaca, "é uma consequência da escolha da política sócio-legal que nós estamos questionando: a decisão de ter proprietários" (STALLMAN, 2002, p. 122).

O ponto crucial para Richard Stallman e os defensores do Projeto GNU é que a sociedade precisa de acesso pleno ao conhecimento, o que é um direito de todos os cidadãos. Essa defesa se baseia na visão "coletivista" do conhecimento que os defensores do projeto têm. O conhecimento é visto como uma propriedade comum e tudo que surge na sociedade, como um novo software, por exemplo, deriva de uma tradição comum. Nesse sentido, o movimento software livre é hoje um importante campo de batalha para garantir o direito de todo cidadão ao livre acesso e ao livre compartilhamento de informações. Para este movimento, a adoção de um sistema aberto e o modelo de software compartilhado refletiria, portanto, um direito e uma responsabilidade social que está acima dos interesses capitalistas individuais.

4 Conclusão

Ao propor uma sociedade na qual todo software é livre, o Projeto GNU força uma reorientação do poder e do conhecimento dentro da configuração social atual, onde o conhecimento na forma de ciência e tecnologia é uma das principais fontes de lucro. Essa reorientação ocorre pela quebra do monopólio de indivíduos ou empresas sobre um bem comum, neste caso o conhecimento tecnológico. Tanto o acesso a esse conhecimento tornado livre e sua reprodução não podem ser controlados por mecanismos de privatização da indústria. O software livre rompe com essa lógica de monopolização e dilui igualmente o poder exercido sobre o conhecimento entre todos os cidadãos, garantindo que o software seja realmente um bem público.

O livre compartilhamento e o trabalho coletivo e cooperativo que ele consequentemente incentiva com o seu modelo de produção são responsabilidades sociais na lógica do movimento. Essa lógica repousa na ideia de manutenção de uma dinâmica comunitária e solidária que é oposta à lógica individualista e competitiva capitalista neoliberal. Nesse sentido, o software livre representa uma rasura no tecido do discurso capitalista e atesta que há alternativas de produção, ao contrário do que quer nos fazer crer o discurso da indústria. O movimento é uma tentativa de resgate de uma lógica de colaboração para que a sociedade prospere com a ajuda de todos, uma chamada para que todos assumam a responsabilidade por essa prosperidade, desempenhando papéis solidários em vez de papéis de dominação e exploração.

Referências

DUMÉNIL, G.; LÉVY, D. Superação da crise, ameaças de crises e novo capitalismo. In: CHESNAIS, F.; DUMÉNIL, G.; LÉVY, D.; WALLERSTEIN, I. Uma nova fase do capitalismo? São Paulo: Xamã, 2003. p. 15-41.

HARDIN, G. The Tragedy of the Commons. Science, v. 162, n. 13, dez. 1968, p. 1243-1248.

HARVEY, D. The new imperialism. New York: Oxford University Press, 2003.

KELTY, C. M. Two bits: the cultural significance of free software. Durham: Duke University Press, 2008.

MARX, K. O capital: Crítica da economia política. Livro I: O processo de produção do capital. São Paulo: Boitempo, 2013.

OSTROM, E. Governing the Commons: The Evolution of Institutions for Collective Action. Cambridge: Cambridge University Press, 1990.

PRADO, E. F. S. Desmedida do valor: Crítica da pós-grande indústria. São Paulo: Xamã, 2005.

STALLMAN, R. M. Free Software, Free Society: Selected Essays of Richard M. Stallman. Boston: GNU Press, 2002.

TORRES, A. L. A tecnoutopia do software livre: uma história do projeto técnico e político do GNU. São Paulo: Alameda, 2018.

WILLIAMS, S. Free as in Freedom: Richard Stallman's Crusade for Free Software. Califórnia: O'Reilly Media, 2002.

---

1 "GNU's Not Unix!" funciona como um acrônimo recursivo para representar a ideia de que o GNU foi baseado no sistema operacional Unix, mas difere dele, principalmente, por ter seu código-fonte aberto.

2 Código-fonte é o conjunto de instruções, em uma linguagem de programação específica, que forma um programa de computador. Se esse código é fechado/proprietário o usuário não consegue acessar essas instruções e não é capaz de descobrir como um determinado programa foi construído, como ele funciona ou, por exemplo, como ele pode ser melhorado.

3 É importante frisar que "livre" nesse contexto não significa grátis ou não comercial. Essa confusão pode acontecer principalmente na língua inglesa, onde a palavra "free" pode significar tanto "livre" quanto "gratuito". O movimento software livre não se opõe à comercialização do software, mas à sua transformação em um bem de acesso restrito através do uso de mecanismos de propriedade intelectual, como o *copyright*.

4 A GPL é uma licença de software livre criada em 1989 e mantida até hoje pela Free Software Foundation, entidade responsável também pela manutenção do Projeto GNU, sistema operacional livre que deu luz a todo esse movimento em defesa de programas com o código aberto.

5 O termo "copyleft" tem sido usado como um trocadilho com *copyright*, que em livre tradução pode significar tanto "deixar copiar", quanto "cópia de esquerda". Ele parece ter surgido em 1984 ou 1985 em uma carta escrita por um amigo de Richard Stallman, que havia escrito a frase: "Copyleft – all rights reversed" (Copyleft – todos os direitos invertidos, em tradução livre) em uma clara referência à frase que acompanha as notificações de *copyright*: "All rights reserved" (Todos os direitos reservados) (WILLIAMS, 2002).

6 Há duas principais correntes dentro do movimento que defende o software livre, uma é chamada de "free software", que é a corrente que defende o caráter social e político do movimento; e a outra é a "open source", que está mais preocupada com questões técnicas e mercadológicas do uso do código aberto. A primeira tem como principal referência a figura do fundador do movimento, Richard Stallman, e a segunda tem como referência Linus Torvalds, criador do Linux, um componente importante dos sistemas operacionais GNU/Linux.

7 Para se aprofundar nessa discussão sobre o software livre como um "commons" ver: SAID VIEIRA, M. What Kind of a Commons Is Free Software? In: CEUR WORKSHOP, Berlim, 2011. Proceedings of the 6th Open Knowledge Conference. Disponível em: https://ssrn.com/abstract=2619956. Acesso em: 20 Maio 2019.

8 É incontestável a importância que esse modelo de produção adquiriu dentro da própria indústria que buscou aboli-lo e adotar em seu lugar o padrão fechado. Isso pode ser atestado pelo investimento maciço de capital e mão de obra que empresas tradicionais têm feito em projetos de software livre, como é o caso da IBM, que só esse ano vai investir 1 bilhão de dólares nesses projetos. Fonte: IBM To Invest $1 Billion In Linux! Disponível em: https://itsfoss.com/ibm-invest-1-billion-linux/. Acesso em: 28 Fev. 2019.