WILKEN DAVID SANCHES

O MOVIMENTO DE SOFTWARE LIVRE E A PRODUÇÃO COLABORATIVA DO CONHECIMENTO

Mestrado em Ciências Sociais

PONTIFíCIA UNIVERSIDADE CATÓLICA – PUC-SP

SÃO PAULO – 2007

WILKEN DAVID SANCHES

O MOVIMENTO DE SOFTWARE LIVRE E A PRODUÇÃO COLABORATIVA DO CONHECIMENTO

Mestrado em Ciências Sociais

PONTIFíCIA UNIVERSIDADE CATÓLICA – PUC-SP

SÃO PAULO – 2007

Dissertação apresentada à banca examinadora da Pontifícia Universidade Católica de São Paulo, como exigência parcial para a obtenção do título de Mestre em Ciências Sociais, sob a orientação da Prf.ª Dr.ª Maria Celeste Mira

1

2 BANCA EXAMINADORA

_____

_____

_____

RESUMO

A dissertação trata da construção de um modelo colaborativo de produção do conhecimento pelo Movimento de Software Livre e seu embate com a atual legislação de propriedade intelectual. É descrita a gênese do movimento de software livre e como ele vem sendo consolidado como um campo autônomo de produção do conhecimento. Para exemplificar o modelo de produção proposto pelo Movimento de *Software* Livre e suas comunidades virtuais, é feita uma análise da organização e da estrutura para tomada de decisões do projeto Debian. A dissertação explora as limitações do atual modelo de propriedade intelectual e como este vem se tornando uma poderosa ferramenta para o aprisionamento do conhecimento dentro de instituições privadas. Por fim, é apresentado, de que forma esse novo modelo de produção colaborativa ultrapassa o desenvolvimento de *softwares* e passa a influenciar outras áreas do conhecimento, abandonando a idéia de propriedade intelectual e aproximando-se do conceito de patrimônio intelectual.

PLAVRAS-CHAVE: *software* livre, comunidades virtuais, campo *hacker, propriedade intelectual, redes de compartilhamento, conhecimento colaborativo, sociedade da informação.*

3 ABSTRACT

This dissertation approaches the construction of a collaborative model of knowledge production disseminated by the Free Software Movement and its opposition to the current law of intelectual property. It describes the genesis of the free software movement as well as how it has become consolidated as an independent domain of knowledge production. In order to better illustrate the model of production proposed by the Free Software Movement and its virtual communities, this paper presents an analysis on the Debian project´s organization and structure for the making of decisions. It explores the limitations of the current model of intelectual property and how it has become a powerful tool for making knowledge a prisoner of private institutions. At last, this dissertation presents how this new model of collaborative production surpasses the software development and begins to influence different areas of knowledge, leaving behing the idea of intelectual property and becoming a concept of intelectual inheritance.

Key words: free software, virtual communities, hacker´s field, intelectual property, sharing nets, collaborative knowledge, information society.

AGRADECIMETOS

Agradeço a todos que estiveram ao meu lado durante este processo auxiliando na coleta de dados e no debate deste tema que tanto me fascina, principalmente à minha orientadora Profa. Dra. Maria Celeste Mira, por sua paciência, perseverança e orientação clara, quando tudo parecia extremamente confuso. Ao amigo Frederico Souza Camara, a quem devo muito pelos longos debates sobre o funcionamento das comunidades de *software* livre e pelas longas horas de consultoria sobre sistemas operacionais e suas estruturas de funcionamento. Aos professores Marco Antonio de Oliveira e Marijane Lisboa, agradeço pelas ponderações e críticas que me auxiliaram a tornar mais consistente esta dissertação. Ao Prof. Dr. Sérgio Amadeu da Silveira por suas ponderações e esclarecimentos sobre o modelo de produção colaborativa e pelo envio de suas pesquisas. Aos amigos Edgard Piccino e Daive Kuhn pela hospitalidade em Porto Alegre, durante as edições do Fórum Internacional do Software Livre e em Brasília. Aos *hackers* Edimilson Novaes, Hugo Cisneiros, José Roberto, Luiz Paulo, Luiz Capitulino, Fernando Ike, Eduardo Maçan, Fábio Telles, Jefferson Alexandre e Eduardo Lisboa pelas entrevistas concedidas, os debates acalorados e principalmente pelo suporte técnico prestado durante toda pesquisa. À Beatriz Tibiriça, João Cassino, Bianca Miguel, Kiminoshin Yoshida, Stela Kuperman, Rodolfo Avelino, Luiz Antônio de Carvalho, Carlos Afonso, Paulo Lima e toda a equipe do Coletivo Digital, Cooperjovem e RITS me apoiaram durante esta jornada. À amiga Milene Ribas, devo muitas horas de debate, apoio na pesquisa e revisão dos textos. Por fim, agradeço aos companheiros do Ácido Lático, Caio Machado, Raul Luiz e Thiago Esperandio, que atrasaram em mais um ano a gravação de seu registro fonográfico para que eu pudesse concluir esta pesquisa.

ÍNDICE

Introdução - Tecnologia e modernidade 08

Capítulo 1 - A gênese do Movimento de *Software* Livre e a Dinâmica do campo *hacker* 20

A dinâmica do campo *hacker* 30

Capitulo 2 - A organização da comunidade Debian 45

Estrutura para a tomada de decisões 56

Capítulo 3 - Conhecimento : proteção versus aprisionamento 74

Capítulo 4 - Da propriedade ao patrimônio intelectual 100

4 Bibliografia 114

5 Sites consultados 117

6 ANEXOS 119

Introdução

Tecnologia e modernidade

O paradigma da modernidade foi criado como um projeto ambicioso e revolucionário, porém repleto de contradições internas. A principal delas é a eterna busca entre o equilíbrio das forças reguladoras e emancipadoras que o compõem.

Durante o processo histórico, cada um desses pilares, sobre os quais foi erguido o paradigma da modernidade, a saber, o pilar da regulação e o da emancipação, teve desenvolvimentos desiguais, acarretando distorções e desequilíbrios entre estas forças.

Segundo Boaventura de Souza Santos (2002), cada um dos pilares de sustentação do paradigma, foi composto por três princípios. O pilar de regulação é formado pelos princípios do Estado, do mercado e da comunidade, e o pilar da emancipação composto pelas lógicas da racionalidade estético-expressiva das artes e da literatura, a racionalidade cognitivo-instrumental da ciência e da tecnologia e a racionalidade moral-prática da ética e do direito. Para que fosse possível a realização do projeto da modernidade, seria necessário alcançar não só o equilíbrio entre os pilares da regulação e da emancipação, mas também o equilíbrio entre os princípios que compunham cada um dos pilares. Contudo, à semelhança dos pilares da modernidade, cada um desses princípios tende à maximização de suas potencialidades o que torna quase impossível o equilíbrio.

Se por um lado a abrangência do projeto da modernidade permitia que se abrisse um vasto campo para as inovações sociais e culturais, sua complexidade interna tornava impossível evitar que algumas de suas promessas fossem cumpridas em excesso, enquanto outras não fossem realizadas. Promessas como erradicação da fome, das guerras, das doenças e do trabalho penoso realizado pelos homens através do uso racional da conhecimento humano, se cumpriram em partes, ou se preferirmos, para uma parte da sociedade.

No campo da regulação, os princípios do Estado, pautados pela obrigação política vertical entre o Estado e os cidadãos e os de comunidade, baseados na obrigação política horizontal solidária entre os membros da comunidade e entre associações foram suprimidos em detrimento do princípio de mercado, o que potencializou ao máximo a obrigação política horizontal individualista e antagônica entre os parceiros de mercado. No campo da emancipação, a lógica potencializada foi a racionalidade cognitivo-instrumental da ciência e da tecnologia. Este desequilíbrio gerou o que Santos definiu como "hipercientificização" do pilar da emancipação, segundo a qual, seria possível por meio da ciência, do uso racional da técnica e da ampliação das capacidades produtivas, acabar com a escassez e a privação, liberando o homem do trabalho penoso, da doença e da fome.

Não há dúvidas de que a modernidade conseguiu cumprir parte de suas promessas. No início do século XXI, boa parte do trabalho penoso dentro das indústrias foi substituído por autômatos. A capacidade produtiva da sociedade teve um aumento sem precedentes e o avanço da ciência na área da saúde permitiu que se descobrisse a cura de várias doenças que no passado assolaram a humanidade. No entanto, algumas promessas ficaram por cumprir. Depois de automatizar as fábricas, a modernidade não soube o que fazer com os trabalhadores que foram substituídos e, apesar de se produzirem alimentos suficientes para todos, a fome no mundo ainda persiste e pessoas ainda morrem de doenças que há muito tempo se conhece a cura.

A hipercientificização do pilar da emancipação permitiu promessas brilhantes e ambiciosas. No entanto, a medida que o tempo passava, tornou-se claro não só que muitas dessas promessas ficaram por cumprir, mas também que a ciência moderna, longe de eliminar os excessos e os déficits, contribuiu para os recriar em moldes sempre renovados, e, na verdade, para agravar alguns deles.(SANTOS – 2002:56)

Durante muito tempo, esses excessos foram interpretados como desvios fortuitos e os défices foram vistos como falhas que seriam corrigidas com o tempo, graças à

aplicação racional dos crescentes recursos materiais e intelectuais que estavam sendo gerados pela modernidade. A correção desses défices e a administração dos excessos da modernidade foram confiados principalmente à ciência e, em segundo, plano ao direito.

No século XIX, a convergência entre o paradigma da modernidade e o capitalismo, e a transformação da ciência em força produtiva fizeram com que os critérios científicos de eficiência e eficácia se tornassem hegemonicos, a ponto de colonizarem os outros princípios emancipatórios. A ética, o direito e arte foram cobertos pela sombra projetada pelos critérios científicos.

Se por um lado, Max Weber viu nesta contaminação das instituições pelos princípios de eficiência científica, um processo de "racionalização" da sociedade, que passava a se expressar dentro das instituições mediante a organização das atividades humanas sob a forma de burocracia, ampliando assim, as esferas sociais que podiam ser submetidas à decisão racional. Por outro lado, Habermas (1994) nos lembra que, o mesmo método científico que nos levou à dominação cada vez mais eficiente da natureza, também forneceu os instrumentos necessários para a dominação dos homens.

Essa racionalidade, estende-se [...], apenas a situações de emprego possível da técnica e exige por isso, um tipo de ação que implica dominação quer sobre a natureza quer sobre a sociedade. A acção racional dirigida a fins é, segundo a sua própria estrutura, exercício de controlos. (HABERMAS – 1994:46)

Com a ciência encarregada de gerir os déficites e excessos da modernidade, suas principais bases de inspiração passaram a ser a utilidade econômica, a efetividade, a velocidade e a funcionalidade do que é pesquisado, e o direito transformou-se em uma muleta do processo de cientificização da sociedade.

Ao direito moderno foi atribuída a tarefa de assegurar a ordem exigida pelo capitalismo, cujo desenvolvimento ocorrera em um clima de caos social que era, em

parte, sua obra. O direito moderno passou assim a constituir um racionalizador de segunda ordem da vida social um substituto da cientificização da sociedade, o ersatz que mais se aproximava – pelo menos no momento – da cientificização da sociedade que só poderia ser fruto da própria ciência moderna. (SANTOS – 2002:119, 120)

Ao se converter em força produtiva, a ciência foi cooptada pelos princípios de mercado e arrastada para o lado das forças de regulação. Esses fatores fizeram com que a ciência assumisse um modelo de desenvolvimento técnico, totalmente atrelado ao capital, conseqüentemente, fazendo com que as pesquisas se voltassem apenas à exploração de mercados e áreas de expansão promissoras. Como ressalta Urich Beck (1999) , as empresas tomam suas decisões sobre seus futuros focos de investigação e de produção sobre a base de prognósticos do mercado.

O problema central é que este tipo de desenvolvimento empresário e de inovação planificada, normalmente, não ocorrem como resposta a sencibilizações sociais e articulações de interesses,[...] portanto, não esta em absoluto influenciado por discussões democráticas sobre as futuras linhas básicas do desenvolvimento técnico e se orienta exclusivamente pela medida dos critérios de rentabilidade e eficiência da economia privada. (BECK – 1999 : 158)

O resultado disso é uma ciência refém da lógica do mercado, pois com exceção das agências estatais de incentivo à pesquisa, a maioria das instituições financiadoras só aportam recursos em projetos que possam gerar lucros à instituição. Mesmo nas universidades estatais brasileiras, já é comum encontrarmos salas, laboratórios e bibliotecas mantidas por multinacionais da indústria farmacêutica, laboratórios de biotecnologia, empresas produtoras de *software* e *hardware*, entre outras, que acabam por influenciar as pesquisas desenvolvidas dentro na Academia em troca da manutenção do espaço e do aporte financeiro feito na universidade.

Com a ciência refém do mercado, desaparece o seu potencial emancipador e

cessa o processo de circulação do conhecimento na sociedade. Hoje, a ciência constitui a principal força produtiva da sociedade, e o conhecimento, sua moeda mais cara.

O resultado é uma sociedade cada vez mais tecnodependente e refém de "sistemas peritos", isto é, "sistemas de excelência técnica ou competência profissional que organizam grandes áreas dos ambientes material e social em que vivemos hoje" e, conseqüentemente, das instituições que dominam estes conhecimentos. (GUIDDENS – 1991 : 35)

À medida que as noções de racionalização, competência e eficiência científicas passaram a contaminar outras esferas sociais, as formas alternativas de conhecimento, ou de produção de conhecimento sobre o mundo passaram a ser desprezadas, legitimando a ciência ocidental como a única forma de saber realmente útil. Vandana Shiva, irá descrever esse processo como a criação das monoculturas do pensamento, que da mesma forma que as monoculturas de plantas importadas, levam à substituição e degradação da diversidade local, destruindo assim, as condições para que hajam formas alternativas de pensar. (SHIVA – 2003 : 25)

A *hipercientificização* do pilar da emancipação, a transformação da ciência em força produtiva e a consolidação das monoculturas da mente trouxeram um aumento sem precedentes da nossa capacidade de ação. Contudo, essa capacidade de ação não veio acompanhada de uma capacidade de previsão das consequências dessas ações, confirmando a afirmação de Santos de que, "a previsão das consequências da ação científica é necessariamente muito menos científica do que a ação científica em si mesma."(SANTOS – 2002:58)

Ao se atrelar a ciência e o conhecimento por ela gerado, à lógica do mercado, tornou-se impossível o desenvolvimento do que Beck chamou de uma "ciência amiga dos erros". Uma vez que, para que a ciência pudesse conviver de forma harmoniosa com seus erros, seria preciso diminuir o ritmo dos avanços tecnológicos, para que estes

erros pudessem ser plenamente compreendidos, de forma que pudéssemos dizer "sim" à técnica e "não" à sua aplicação, quando este fosse o caso. (BECK – 1999 : 159)

Uma ciência "amiga dos erros" permitiria que laboratórios pesquisassem sobre variedades de vegetais geneticamente modificados, e ao final de sua pesquisa, caso fosse constatado o risco de contaminação ou mesmo a falta de capacidade de prever tais riscos, dissessem não à aplicação daquela técnica. Se pensarmos no campo da informática, que a cada dia se torna mais presente em nossas vidas, uma ciência amiga dos erros poderia chegar à conclusão que o *up-grade* de um determinado software, apesar de gerar alguns minutos de economia ao realizar determinada operação, necessitaria de uma renovação do parque de máquinas, que geraria uma grande quantidade de lixo técnológico, composto em grande parte por material tóxico, que a longo prazo traria grande prejuízo ao meio ambiente. Nesta situação, novamente poderiamos dizer "sim" à tecnica e "não" à sua aplicação.

Quando se afirma que somos reféns dos sistemas peritos, é justamente por não existir um debate com a sociedade, nem instâncias adequadas para que tal debate aconteça; a fim de que se decida sobre a aplicação ou não de uma determinada técnica, essas decisões são delegadas aos peritos, que são responsáveis por calcular os níveis aceitáveis de risco.

Santos (2002) apresenta duas leituras possíveis dessa incapacidade de prever as conseqüências da ação científica: a primeira dirá que a capacidade de ação da ciência é excessiva em relação à sua capacidade de previsão das conseqüências, enquanto a segunda afirmará que a capacidade de previsão das conseqüências das ações científicas se encontra deficitária. Essas duas leituras apresentam interpretações radicalmente opostas, já que a primeira põe em questão a noção de progresso científico, enquanto a segunda se limita a exigir mais progresso. Contudo, à medida que os riscos assumidos individualmente pelos técnicos e pelas empresas se tornam riscos coletivos, que por sua vez geram lucros privados, ganha credibilidade a interpretação que põe em questão

a noção de progresso científico e as formas alternativas de pensamento voltam a ter voz.

Acredito que estamos vivendo um momento de transição no qual o atual modelo científico e seus valores estão sendo postos em questão; os riscos eminentes de contaminação do solo, da água e do ar, o perigo constante de guerras fazem com que a sociedade procure uma alternativa, que não virá por meio da negação da ciência e da técnica, mas da retomada de seu potencial emancipador. Para que seja possível gerar uma forma alternativa de pensamento, será necessário desvincular a ciência da lógica do capital, a fim de que possamos gerar conhecimento como ferramenta de emancipação, garantindo, assim, sua livre circulação em vez de aprisioná-lo dentro das grandes corporações.

Uma mudança de tal envergadura passa necessariamente pela reestruturação do estatuto jurídico da ciência, que como vimos foi formulado com o intuito de garantir a ordem social necessária para o capitalismo e conseqüentemente para a consolidação da ciência como sua principal força produtiva.

Como veremos mais adiante, as patentes são hoje a forma mais poderosa de perpetuação do atual modelo de produção científica, que impede a circulação do conhecimento mediante a proteção dos direitos de propriedade intelectual e aprisiona o conhecimento dentro de instituições privadas. Esse tipo de legislação vem se configurando como a principal ameaça à comunicação entre os cientistas envolvidos em empreendimentos comerciais protegidos por elas. Segundo a afirmação do nucleobiólogo Emanuel Estein:

No passado trocar idéias irrefletidamente era a coisa mais natural do mundo entre colegas para compartilhar as idéias recém saídas do contador de cintilografia ou de uma célula de eletroforese, para mostrar uns aos outros esboços de artigos e dessa forma agir como companheiros numa pesquisa cheia de entusiasmo. Isso já não

acontece mais. Qualquer cientista da UCD (Universidade da California em Davies) com uma nova e promissora cultura para (o melhoramento da safra)... há de pensar duas vezes antes de falar sobre isso com qualquer uma das duas empresas privadas de genética de plantas agrícolas de Davies – ou mesmo com colegas que por sua vez, poderiam falar com essas pessoas. Eu sei que esse tipo de inibição já esta acontecendo. (KENNETH apud SHIVA – 2001:36, 37)

Nos capítulos que se seguem, será demonstrado como as atuais leis de propriedade intelectual, que foram criadas como um reflexo do paradigma da modernidade e com o argumento de proteger e incentivar o avanço da ciência, estão se consolidando como a principal ferramenta para o fechamento do conhecimento dentro de instituições privadas, promovendo assim, a ampliação da capacidade reguladora da ciência. Além disso, serão apresentadas a gênese e a estrutura de funcionamento do Movimento de *Software* Livre, que vem consolidando uma nova forma de produção tecnológica, que impede a apropriação privada do conhecimento, garante a sua livre circulação na sociedade e aos poucos começa a contaminar outras áreas de conhecimento.

Contudo, é importante ressaltar que a modernidade ocidental não pressupunha o capitalismo como o sistema de produção; ambos são processos distintos e autônomos. Na verdade, o socialismo marxista, ou socialismo científico, tal como o capitalismo, é parte constitutiva da modernidade. Durante a Guerra Fria, tanto capitalistas como comunistas mediam o grau de sucesso de cada um dos lados de acordo com o volume da produção industrial e com os avanços científicos por eles conquistados. É justamente esta disputa, que irá gerar as bases da revolução informacional em que nós vivemos e a infra-estrutura necessária para a consolidação do Movimento de *Software* Livre, como uma voz dissidente, dentro do paradigma da modernidade.

Um dos mais importantes marcos da corrida tecnológica gerada pela Guerra Fria data de 1957, quando a antiga URSS lançou o primeiro satélite espacial, o Sputinik,

dando assim o pontapé inicial à corrida que originou um ciclo de revoluções tecnológicas sem precedentes. Isso porque, em resposta à façanha soviética, o então presidente dos Estados Unidos da América, Eduight Eisenhower, criou a ARPA (Advanced Research Projects Agency), que iniciou uma série de projetos ousados, visando retomar a liderança norte-americana na disputa tecnológica. Um desses projetos foi a criação de um sistema de comunicação invulnerável ao ataque nuclear, idealizado por Paul Baran e pela Rand Coporation.

Com base na tecnologia de comutação de pacotes, o *packet switching*, pretendia-se criar um sistema de comunicação independente de centros de controle, mediante o qual as mensagens seriam quebradas em pequenas partes, e depois de enviadas, encontrariam suas rotas ao longo de uma rede, sendo remontadas com sentido coerente, em qualquer ponto dela.

Em 1968, Larry Roberts, Ivan Sutherland e Bob Taulor, da ARPA, organizam quatro universidades americanas para a implantação da rede de comutação de pacotes. Denominada ARPANET, a rede de comunicação unia inicialmente apenas as universidades de Stanford, Berkley, UCLA e Utah. Aberta, a princípio, apenas para estes poucos centros de pesquisa, a ARPANET começou a se desviar de seu objetivo inicial assim que os cientistas começaram a utilizar a rede, para trocar todo o tipo de informação. Rapidamente, ficou difícil de separar as mensagens com fins científicos e militares das correspondências pessoais.

Esse desvio na função inicial da rede foi tamanho, que em 1983 ocorreu a divisão da ARPANET, dedicada para fins científicos e a MILNET, dirigida a fins militares. Paralelamente, a Fundação Nacional da Ciência também passou a se dedicar à criação de uma rede científica de comunicação, a CSNET, e uma rede para fins não científicos, a BITNET. Todas essas redes, no entanto, ainda utilizavam a ARPANET como sistema de comunicação, que passou então a ser denominada ARPAINTERNET e, posteriormente, INTERNET. (CASTELLS 1999: 375; 376)

As conquistas tecnológicas das décadas de 50, 60 e 70, do século XX, tais como o avanço da microeletrônica, a inveção do transistor, do microprocessador e a popularização da computação pessoal, fundam o que Manuel Castells definiu como a sociedade em rede. Munida desses avanços tecnológicos, mais uma vez, a modernidade reinvetou em novos moldes, antigas desigualdades, criando outras possibilidades de regulação da humanidade. Contudo, algumas características do novo modelo abriram outras possibilidades de emancipação, atualmente exploradas pelos militantes do Movimento de *Software* Livre.

Algumas dessas características emancipatórias nascem junto com a própria Internet, que se configurará como o principal veículo de comunicação e articulação desses novos ativistas, que encontram, no barateamento da computação pessoal e na comunicação em rede, uma forma alternativa de produção do conhecimento, aqui designada como produção colaborativa.

Como vimos anteriormente, a Internet tem seu impulso inicial por meio da ação combinada da grande ciência e os interesses militares do governo norte americano. No entanto, é pela interface entre os programas de macropesquisa e grandes mercados desenvolvidos pelos governos e a inovação descentralizada estimulada por uma cultura de criatividade tecnológica e por modelos de sucessos pessoais rápidos, que a tecnologia da informação proposperará.

Ao tratarmos a história da sociedade da informação, temos que ter em mente as três forças que a impulsionam e modelam seu desenvolvimento, a saber, as ações e iniciativas de macropesquisa governamentais, muitas vezes com fins militares, as inovações tecnológicas levadas a cabo pelo mercado e a ação dos *hackers*. Se por um lado a ARPA deu o impulso inicial ao que viria a se tornar a Internet, ao financiar um projeto de criação de uma rede de comunicações horizontal, foram os *hackers* que possibilitaram a conexão entre os microcomputadores e mais adiante a conexão desses à Internet por meio de uma linha telefônica comum.

O Modem, importante elemento desse sistema, foi inventado por dois estudantes de Chicago, Ward Christensen e Randy Suess, em 1978, quando tentavam transferir programas de um microcomputador para outro utilizando uma linha telefônica, a fim de evitar uma longa viagem entre suas localizações durante o inverno de Chicago. Em 1979, os inventores difundiram o protocolo Xmodem, permitindo que computadores transferissem arquivos diretamente sem passar por um sistema principal. E difundiram a tecnologia sem nenhum custo porque o objetivo era espalhar as capacidades de comunicação o máximo possível.(CASTELLS – 1999:337)

Os *hackers* fazem parte da história dos intelectuais que desenharam o atual formato da Internet, sem necessariamente receber financiamentos do Estado, ou o apoio das universidades, mesmo quando se encontravam dentro de sua estrutura. A ação desses agentes buscava maximizar o potencial emancipador das novas tecnologias, não só por meio do desenvolvimento de novas técnicas, mas também propondo novos usos ao conhecimento já existente.

Atualmente, o termo *hacker* passou a ser utilizado, de forma equivocada, pela imprensa para rotular pessoas que cometem crimes utilizando-se de recursos informacionais. No entanto, quando me referir ao termo *hacker* durante essa dissertação, estarei fazendo referência ao seu significado original que está ligado às idéias de "programar por prazer" e do "fazer elegante", as quais serão retomadas mais adiante. Para descrever os indivíduos que utilizam seus conhecimentos para cometer crimes através do uso de computadores, o termo adequado é *cracker*.

O que começou com ações desconexas nos anos 60 e 70 acabou por dar subsídio ao que viria a se tornar uma espécie de ética *hacker*. No início dos anos 80, começam a surgir as primeiras ações do Movimento de *Software* Livre, que é hoje o grande campo de atuação dos *hackers* em nossa sociedade, apesar de não ser o único. No capítulo seguinte, serão apresentados alguns termos que serão utilizados durante toda esta

dissertação, além de dar um breve panorama sobre a história e o funcionamento do Movimento do *Software* Livre e a forma como este tenta se consolidar como um campo autônomo de produção do conhecimento, tal qual o campo da arte ou da produção erudita.

7 Capítulo – 1

8 A gênese do Movimento de *Software* Livre e a Dinâmica do campo *hacker*

Um dos eventos fundadores do Movimento de Software Livre foi, sem dúvida, a compra de uma impressora da marca Xerox pelo Laboratório de Inteligência Artificial do MIT (*Massachusetts Institute of Technology*), onde trabalhava Richard Stallman, considerado um dos principais ideólogos do movimento. Mais importante do que a compra do novo equipamento, o Movimento de *Software* Livre foi beneficiado por um defeito do *driver* de dispositivo da impressora, isto é, do programa de computador que gerenciava seu funcionamento. Depois de fracassarem todas as tentativas para colocar a nova impressora em funcionamento, a equipe resolveu contactar a Xerox e pedir o código fonte[1] do programa que a gerenciava, para que assim fossem feitas as mudanças necessárias e ela passasse a funcionar no computador do MIT. A Xerox forneceu o código fonte e as alterações foram feitas com sucesso. Procurados por outra universidade, que também havia comprado uma impressora do mesmo modelo, Stallman e seus companheiros de equipe forneceram o código fonte com as alterações necessárias, para o funcionamento do equipamento. No ano seguinte, o MIT renovou seu parque de máquinas e outra impressora da Xerox foi adquirida. Como já havia acontecido no passado, o equipamento não funcionou e novamente Stallman procurou a Xerox para que ela fornecesse o código fonte do programa de gerenciamento da impressora. No entanto, dessa vez a Xerox negou-se a fornecer o código, a menos que a equipe do MIT pagasse por ele e assinasse um *"nondisclosure agreement",* um acordo de não revelar. Stallman não aceitou o acordo e resolveu procurar a universidade a qual

1 O Código Fonte é o conjunto de instruções que são passadas ao computador para que ele execute uma determinada tarefa. O software é o resultado dessas instruções. Aqui podemos traçar um paralelo entre se fazer um bolo e um programa de computador, no caso do Bolo a receita será o seu Código Fonte, é o acesso à receita que permitirá que o cozinheiro possa adaptar e melhorar o bolo, acrescentando ou substituindo os ingredientes.

eles haviam ajudado no passado. A equipe da universidade disse ter enfrentado as mesmas dificuldades com o novo equipamento, mas havia solucionado os problemas com uma alteração no código fonte do programa. No entanto, quando Stallman solicitou o código com as alterações, os desenvolvedores lhe disseram que não poderiam fornecer, pois haviam assinado um *nondisclosure agreement.*

Em uma de suas palestras sobre o projeto GNU, realizada em fevereiro de 2002, no 2º Fórum Social Mundial, Stallman disse ter ficado tão inconformado com aquela situação que decidiu que não seria mais vítima, nem vitimaria outras pessoas com acordos de *nondisclosure agreement.* Quando ingressou no MIT em 1971, Stallman fazia parte de uma equipe que utilizava exclusivamente *softwares* livres. Esta equipe estava inserida em uma comunidade maior, que agregava equipes de vários laboratórios e universidades, que estavam acostumados a trocar informações e programas de computador como uma forma de se ajudarem mutuamente. Contudo, esse universo de compartilhamento do conhecimento estava acabando, com a mudança do modelo de negócio das empresas de *software* nos anos 80, que passaram a cobrar pela licença de uso de seus programas. À medida que aumentava o número de laboratórios que utilizavam *softwares* proprietários, a comunidade na qual Stallman estava inserido passava a desaparecer. (Stallman-2005:161)

Em 1983, Richard Stallman lança o manifesto intitulado UNIX Livre. O documento anunciava a criação de um projeto chamado GNU (GNU is not UNIX), um acrônimo recursivo utilizado para dizer que o GNU seria um sistema operacional livre, compatível com o UNIX, mas não idêntico.

Em 1984, Richard Stallman renuncia ao seu cargo no MIT e começa a programar o Sistema GNU. Com sua saída do MIT, Stallman garante que a instituição não possa influenciar na distribuição do sistema GNU como um *software* livre e evita que seu trabalho se desvie da finalidade para a qual foi idealizado. (Stallman – 2005:164)

Em 1985, é fundada a FSF (*Free Software Foundation*) uma instituição sem fins lucrativos com o objetivo de captar recursos, estimular o desenvolvimento e disseminar o uso do *software* livre, além de dar respaldo jurídico aos seus desenvolvedores. Nesse momento, o conceito de *software* livre consolida-se e, a partir de agora, quando me referir ao termo, terei como base a definição da *Free Software Foundation*, segundo a qual, para que um programa de computador possa ser definido como um software livre ele deverá garantir ao usuário quatro liberdades, a saber:

1. liberdade de executar o programa para qualquer finalidade, sem qualquer tipo de embargo;

2. liberdade de alterar o programa, para que ele se adapte às suas necessidades. (Para que esta liberdade seja assegurada, é indispensável que o programa venha acompanhado do seu código fonte);

3. liberdade de distribuir cópias do programa, gratuitamente, ou em troca de um pagamento;

4. liberdade de distribuir cópias modificadas para que toda a comunidade possa se beneficiar das melhorias feitas no programa.

O projeto GNU foi baseado em grande parte no sistema operacional criado pelo Bell Labs da AT&T, o UNIX. Mas por que optar por fazer um sistema semelhante ao UNIX ao invés de criar um sistema operacional do marco zero?

Para que se possa entender essa estratégia, é preciso ter claro o que é um sistema operacional e como se deu o surgimento do próprio UNIX. qual o motivo desse sistema, e não outro, tornar-se o preferido principalmente entre os laboratórios das universidades, passando a ser o principal sistema instalado nos supercomputadores dessas instituições?

Comecemos então pela definição de sistema operacional. Deitel nos traz a

seguinte definição:

Vemos um sistema operacional como os programas implementados como software ou firmware, que tornam o hardware utilizável. O hardware oferece a capacidade computacional bruta. Os sistemas operacionais disponibilizam convenientemente tais capacidades aos usuários, gerenciando cuidadosamente o hardware para que se obtenha uma performance adequada(DEITEL – 1992 apud JANDL – 1999:10)

A definição de Deitel nos apresenta novos termos a serem esclarecidos como *hardware*, *software* e *firmware*. O *hardware* são as partes físicas do sistema computacional, isto é, o conjunto de dispositivos elétricos, eletrônicos, ópticos e eletromecânios que compõem o computador. Os *softwares* são os programas que estão instalados no computador e representam a parte lógica do sistema computacional. Por fim, o *firmware* é representado por programas especiais armazenados de forma permanente no *hardware* do computador que permitem o funcionamento elementar e a realização de operações básicas em certos dispositivos do computador, tais como placas de rede, ou associados a alguns periféricos, como impressoras, roteadores e scaners.

O firmware geralmente vem acondicionado em circuitos de memória não volátil (ROM, PROM e EPROM) sendo os programas ali gravados escritos geralmente em linguagem de máquina e destinados à execução de operações especiais tal como a auto-verificação inicial do sistema e a carga do sistema operacional a partir de algum dispositivo adequado. (JANDL – 1999:10)

Para que um computador se torne utilizável além, dos dispositivos de *hardware* são necessárias várias camadas de *software,* o sistema operacional constitui apenas uma delas, como podemos ver no diagrama abaixo:

O sistema operacional envolve toda a camada física do computador sendo o responsável por intermediar a comunicação entre o usuário e o *hardware* do computador. Caso ele não existisse, todas as solicitações que fossem feitas à máquina deveriam ser emitidas em linguagem binária, isto é, em forma de 0 e 1, o que transformaria a mais simples das aplicações em uma penosa tarefa. Além disso, o sistema operacional também fornece um ambiente de desenvolvimento mais apropriado para que possam ser confeccionados os chamados aplicativos, os softwares para tarefas específicas, como editores de texto, planilhas de calculo, clientes de *e-mail* que rodam sobre o sistema operacional. Mais uma vez, caso não houvesse um intérprete entre o homem e a máquina, os aplicativos deveriam ser escritos todos em linguagem de máquina, isto é, comandos binários, o que tornaria essa tarefa ainda mais difícil, uma vez que existem várias arquiteturas de computadores, cada uma com uma linguagem específica.

Devido ao grau de complexidade envolvida no desenvolvimento desses sistemas, eles são formados por várias camadas e vários programas menores, além de um núcleo central, denominado *kernel*, responsável por gerenciar prioridades e garantir a comunicação entre os outros programas que formam o sistema operacional.

Retomemos agora o motivo pelo qual o UNIX foi utilizado como o modelo a ser seguido para o desenvolvimento do projeto GNU. Quando lança o Manifesto GNU (ANEXO1), Richard Stallman inicia o seu texto com o seguinte parágrafo:

GNU, que significa Gnu Não é Unix, é o nome para um sistema de software completo e compatível com o Unix, que eu estou escrevendo para que possa fornecê-lo gratuitamente para todos os que possam utilizá-lo. Vários outros voluntários estão me ajudando. Contribuições de tempo, dinheiro, programas e equipamentos são bastante necessárias.(STALLMAN - 1983)

Em outro trecho do manifesto, que também merece destaque para a nossa análise, Stallman diz o seguinte:

GNU será capaz de rodar programas do Unix, mas não será idêntico ao Unix. Nós faremos todos os aperfeiçoamentos que forem convenientes, baseados em nossa experiência com outros sistemas operacionais... Unix não é o meu sistema ideal, mas ele não é tão ruim. Os recursos essenciais do Unix parecem ser bons recursos, e eu penso que posso fornecer o que falta no Unix sem comprometê-lo. E um sistema compatível com o Unix seria conveniente para muitas pessoas adotarem.(STALLMAN - 1983)

Na verdade, a escolha por fazer um sistema baseado no UNIX facilitou o trabalho colaborativo dentro do projeto GNU. Se o sistema fosse ser criado do marco zero e cada desenvolvedor interessado no projeto programasse uma parte diferente do sistema, isso acarretaria uma demora muito grande para se obter os primeiros resultados práticos, além de um esforço hercúleo com o propósito de juntar as diferentes partes do sistema. Ao passo que se cada desenvolvedor confeccionasse um programa equivalente a um do UNIX e compatível com esse sistema operacional, o trabalho seria muito mais fácil, não só para os desenvolvedores, como para as pessoas interessadas em utilizar o novo sistema

operacional. A idéia por trás do projeto GNU é que, assim como o UNIX, ele seria um sistema modular. Dessa forma, a medida que novos programas ficassem prontos eles poderiam substituir os seus equivalentes no UNIX, até que um dia, todo o sistema fosse GNU.

Outro motivo para a escolha do UNIX foi sua popularidade no meio acadêmico, que está diretamente ligada à forma como o sistema foi concebido. O UNIX foi desenvolvido por Dennis Riche e Ken Thompson no Bell Labs da AT&T e rapidamente ganhou os laboratórios das universidades adquirindo grande popularidade entre os *hackers*. Parte desse sucesso se deu pela sofisticação do sistema que apresentava a filosofia do "pequeno é bonito", como proposta de funcionamento, contendo um pequeno conjunto de blocos de construção simples e que podiam ser combinados para obter resultados de alta complexidade. Em seu livro *Só por prazer*, Linus Torvalds (2001) lembra que a simplicidade do UNIX não pode ser confundida com a falta de sofisticação dando como exemplo os idiomas humanos.

O inglês tem 26 letras e é possível fazer tudo a partir delas. E temos o chinês, em que há uma letra para cada coisa que se possa pensar. Nele você parte da complexidade e pode combina-las de forma limitada[...] a escrita pictória como os caracteres chineses e os hieróglifos, tende a acontecer primeiro e a ser "mais simples", ao passo que o enfoque do bloco de construção demanda muito mais pensamento teórico. (TORVALDS – 2001: 80, 81)

Além da sofisticação e robustez do sistema, a popularidade do UNIX se deu em boa parte pela maneira como foi desenvolvido e disponibilizado aos usuários. Se, por um lado, o requinte teórico saltava aos olhos quando se analisava a estrutura de funcionamento do UNIX, por outro, os objetivos que motivaram seus criadores durante a confecção do sistema podem não parecer tão sofisticados assim. Na verdade, o sistema foi criado originalmente para que seus desenvolvedores pudessem jogar Guerra nas Estrelas em um computador PDP-11. (TORVALDS – 2001: 81)

O UNIX começou como um projeto pessoal de Denis Richie e Ken Tompson e, como o sistema operacional não era considerado um projeto sério, a AT&T não pensou nele em termos comerciais. Assim, seus criadores o disponibilizaram livremente, junto com seus códigos fonte, principalmente para universidades.

É preciso lembrar que a AT&T era um monopólio controlado pelo Estado e, até 1984, não tinha autorização para comercializar programas de computador. No entanto, quando houve a cisão da AT&T e ela obteve a permissão para ingressar no mercado de *softwares*, a empresa tentou retomar o UNIX como um produto comercial. O problema encontrado pela AT&T é que nessa época os cientistas da computação das universidades, em particular os da Universidade da Califórnia em Berkeley, já vinham trabalhando nele há anos.

Em 1990, o UNIX já havia se tornado o sistema operacional número um nos computadores de grande porte e representava um negócio de milhões de dolares, o que fez com que a AT&T acionasse judicialmente a Universidade da Califórnia em Berkeley. O objetivo da ação era impedir que a universidade continuasse mantendo e distribuindo uma versão do UNIX chamada de BSD (*Berkeley Software Distribution*), uma vez que o código original pertencia à AT&T. A universidade, por sua vez, afirmava que o código da AT&T tinha sido quase todo reescrito graças ao esforço de seus cientistas, e que a empresa não tinha o direito de impedir que a universidade distribuísse gratuitamente, ou mesmo cobrasse um valor nominal sobre as cópias do UNIX BSD, por ela mantido. (TORVALDS – 2001: 82)

Esse impasse judicial só teve fim quando a Novell Inc. comprou o código fonte do UNIX da AT&T e retirou o processo contra a Universidade da Califórnia em Berkeley. Durante o período, em que o impasse se arrastou na justiça, novos atores entraram em ação e começaram a ganhar espaço junto aos usuários do UNIX; eram eles o próprio projeto GNU e o Linux, que mais tarde se fundiriam no sistema operacional GNU/Linux.

Essa fusão se deu por um atraso no projeto GNU. Como disse anteriormente, um sistema operacional é formado por vários programas menores, além de um núcleo central denominado *kernel,* que tem por função garantir a comunicação entre todos esses programas e gerenciar as prioridades do sistema. No início do projeto GNU, Stallman pretendia que o sistema funcionasse com um esquema de *microkernels*, ao invés de um *kernel* monolítico que gerenciaria todo o sistema operacional. Ele teria vários núcleos que dividiram as tarefas em quantas partes fossem possíveis, resolveria cada uma delas e depois as juntaria novamente, tal qual o método cartesiano. Dessa forma, ele pretendia diminuir o grau de complexidade das tarefas executadas e aumentar a performance do sistema. Contudo, um atraso na conclusão do *kernel* do GNU faz com que, em 1991, Linus Torvads lance a primeira versão do Linux, um projeto de *kernel* monolítico que se utilizava de alguns programas já desenvolvidos pelo projeto GNU. Ao contrário de Stallman, Linus Torvalds achava que um *kernel* monolítico, que controlasse todo o sistema operacional seria muito mais eficiente e simples de programar, pois, depois de quebrar todas as tarefas executadas pelo sistema operacional em centenas de partes e resolvê-las separadamente, a tarefa de junta-las novamente acabava sendo extremamente complexa e prejudicava ainda mais a performance do sistema operacional. Ainda hoje existem os defensores do sistema baseado em *microkernels,* e esse subprojeto não foi interrompido dentro do projeto GNU. Apesar de ter um ritmo de desenvolvimento muito lento, o *Hurd*, como é chamado o projeto de *microkernels* do GNU, continua sendo aperfeiçoado. Mas o fato é que Torvalds tinha razão em pelo menos um ponto, o *kernel* monolítico era muito mais fácil de programar e, por esse motivo, ele foi o primeiro a ficar pronto.

Em 1993, Patrick Volkerding lança o Slackware à disttribuição GNU/Linux, mais antiga ainda em atividade, que significa sistema GNU com *kernel* LINUX. Daí em diante, várias outras distribuições seriam lançadas, algumas mantidas por universidades, outras desenvolvidas por empresas e outras que contavam apenas com a

colaboração de *hackers* espalhados pelo mundo e que tinham como principais ferramentas de ação seus conhecimentos e um computador conectado à Internet.

Com a evolução do s*oftware* livre, as distribuições passaram a funcionar da mesma forma que as editoras, tendo como função publicar, no sentido de tornar público, e homologar a qualidade do que é desenvolvido pelas comunidades de *software* livre, tornando-se também as principais entidades legitimadoras do conhecimento *hacker*. Assim como os editores, os mantenedores de uma distribuição são os responsáveis por sondar quais projetos estão sendo desenvolvidos pelas comunidades e qual o nível de estabilidade de cada programa, para que possam fazer uma coletânea de programas, empacota-los em uma mídia e distribuí-los aos interessados. Um usuário que adquirir um CD de uma distribuição encontrará nele, além dos pacotes básicos para o funcionamento do sistema operacional GNU/LINUX, todos os programas e aplicativos que os mantenedores da distribuição acham necessários para o funcionamento de um computador, como editores de texto, imagem, vídeo e som, planilhas eletrônicas, navegadores de Internet, entre outros.

À medida que o *software* livre ganha mais espaço, a organização do movimento passa a adquirir um grau maior de complexidade e começam a se formar diversas comunidades em torno de distribuições e de *softwares* específicos, com o intuito de acelerar o seu desenvolvimento. Além disso, passa a ser criada uma série de organizações não governamentais e grupos de usuários, cujos objetivos variavam desde levantar fundos para os projetos e dar respaldo jurídico aos desenvolvedores, até difundir o *software* livre em outros campos da sociedade. Assim, de agora em diante quando me referir ao Movimento de *Software* Livre, estarei fazendo menção a todas essas instituições. O termo comunidade será utilizado para designar os grupos de programadores, ou usuários que se organizam em torno de um projeto específico ou de uma distribuição.

A dinâmica do campo *hacker*

No início do movimento, as comunidades de *software* livre podiam ser comparadas a uma "sociedade de admiração mútua". A maioria dos desenvolvedores se conheciam por intermédio das listas de discussão na Internet, das quais faziam parte, além do que, não existia uma preocupação em estabelecer qualquer tipo de comunicação com as pessoas que estavam fora deste círculo. Os programas que eram desenvolvidos para o GNU/LINUX não tinham a preocupação em atender o usuário final, isto é, o não programador, o que, no início, fez com que a maioria de seus softwares não tivessem ambiente gráfico e funcionassem apenas no modo texto. Também não havia uma grande preocupação em portar os *softwares* para as diferentes arquiteturas de computador que existiam naquela época; a idéia era que cada um fizesse as customizações necessárias para sua arquitetura e as disponibilizassem na Internet para os outros usuários. Talvez o título do *e-mail* postado por Linus Torvalds na lista de discussões do Minix, um sistema operacional da época, para anunciar o lançamento da versão 0.02 do Linux seja um dos grandes exemplos do grau de especialização do público a que se destinava este tipo de programa. Dizia ele: *Você suspira pelos dias em que os homens eram homens e escreviam seus próprios drivers de dispositivo?.(*Torvalds – 2001 : 117)

Segundo Pierre Bourdieu:

pode-se medir o grau de autonomia de um campo de produção erudita com base no poder que dispõe para definir as normas da sua produção, os critérios de avaliação de seus produtos e, portanto para retraduzir e reinterpretar todas as determinações externas de acordo com seus princípios de funcionamento (BOURDIEU – 2004 : 106),

E foi dessa forma que passou a agir o Movimento de *Software* Livre. Esse processo de autonomização do movimento aconteceu atrelado à formação de um grupo

de intelectuais e *hackers*, cada vez mais inclinados a levar em consideração apenas os quesitos técnicos e as regras internas do Movimento de *Software* Livre em suas tomadas de decisão e cada vez mais propensos a libertar sua produção intelectual das regras mercadológicas ditadas pela indústria do *software* da época.

Os desenvolvedores ligados ao Movimento de *Software* Livre se organizam em torno de projetos, quase sempre trabalhando como voluntários durante seu tempo livre. À primeira vista, a auto-organização do movimento pode parecer um tanto quanto caótica, já que cada desenvolvedor se junta ao projeto com o qual tem maior afinidade, o que faz com que alguns centrem esforços no *kernel* do sistema operacional, isto é, no Linux, e outros dividam-se entre os milhares de subprojetos do projeto GNU, desenvolvendo e aperfeiçoando as outras partes do sistema operacional e os aplicativos que rodam sobre ele, além de outros projetos que não estão dentro do escopo do sistema GNU, mas que estão sobre alguma licença de uso que o torne um software livre.

Mas como funciona essa engrenagem que agrega milhares de colaboradores ao redor do mundo, sem que haja uma estrutura central organizando o trabalho?

Pois bem, para cada programa, temos um mantenedor que, em geral, é a pessoa que deu início ao trabalho e que publicou pela primeira vez seu código fonte e a documentação inicial do *software*. Abaixo do mantenedor ou mantenedores, estão os desenvolvedores que se agregam ao projeto de acordo com seu grau de interesse e conhecimento.

Para que possamos entender melhor esse mecanismo, tomemos como exemplo um projeto fictício de um novo editor de textos, a pessoa ou o grupo de pessoas que iniciar o projeto torna-se automaticamente seu mantenedor. Para disponibillzá-lo para a comunidade e agregar mais desenvolvedores a sua volta, os mantenedores devem disponibilizar a primeira versão do software na Internet, juntamente com seu código

fonte e com a documentação inicial que deve conter as informações detalhadas de como a primeira versão foi feita, qual o objetivo final do *software* e quais funcionalidades ele pretende ter. O ato de disponibilizar o *software* na Internet pode ser feito por meio de um site próprio do projeto, em um site que agregue vários projetos, a exemplo do www.sourceforce.org ou em ambos, como é mais comum.

Depois que o *software* foi publicado, desenvolvedores que têm interesse em produzir um editor de texto com as mesmas funcionalidades e objetivos que o editor do nosso exemplo irão se agregar ao projeto, baixando o programa da Internet, juntamente com a documentação e o código fonte. Durante seu tempo livre, esses programadores irão realizar modificações e acréscimos ao código fonte original, que serão enviadas ao mantenedor do projeto, cuja função será analisar as modificações, correções e acréscimos e julgar se elas são pertinentes ao projeto. Caso essas alterações sejam consideradas úteis, serão integradas à próxima versão do *software* que for disponibilizada aos usuários, e os nomes dos programadores que contribuíram poderão ser colocados na lista de autores do programa, em uma parte específica do código fonte.

No caso das modificações que não são integradas ao projeto, elas podem seguir três caminhos. O primeiro é serem completamente esquecidas, a segunda opção é que elas sejam utilizadas apenas pelo programador, ou grupo de programadores que as desenvolveram, o que é muito comum. O terceiro caso só costuma acontecer em projetos de grande envergadura no qual as modificações que são rejeitadas poderiam levar o projeto para um outro lado, que não o de seu objetivo inicial, isto é, modificações que poderiam redefinir o escopo do projeto; nesses casos são criados *forks* ou divisões do projeto. Voltemos ao nosso exemplo do editor de textos. Digamos que o projeto inicial seja o de criar um editor do tipo bloco de notas, que irá trabalhar unicamente com texto puro, isto é, arquivos do tipo ".txt" e ele receba uma contribuição que pretende incluir a possibilidade de se trabalhar com tabelas dentro do

programa, e que esta modificação acarrete na criação de uma nova extensão para os arquivos gerados dentro do nosso *software*, e que essa extensão ".xyz" só poderá ser manipulada dentro do nosso editor de textos. Caso o mantenedor julgue a contribuição desnecessária, poderá simplesmente ignorá-la. No entanto, os autores poderão gerar um *fork* do projeto, que utilizará todo o desenvolvimento do editor original, acrescentando as alterações que permitem o uso de tabelas. O *fork* seguirá o mesmo processo de divulgação na internet e em seu código fonte, deverá constar que ele consiste em um projeto gerado a partir de um *fork* do editor original, juntamente com os nomes dos autores que haviam contribuído para o programa original até a data da divisão.

Quando acontece um *fork*, o que irá definir qual dos dois projetos terá mais sucesso, será o grau de distinção dos mantenedores dentro do campo *hacker* e as características técnicas dos projetos. A noção de campo representa para Bourdieu(2004) um espaço social de dominação e de conflitos. Cada campo tem uma certa autonomia e possui suas próprias regras de organização e de hierarquia social. Como veremos adiante, o campo *hacker* é onde o conhecimento colaborativo convive com a competitividade e a luta pela legitimidade tecnológica de seus membros, que buscam, cada vez mais, alcançar distinções culturalmente pertinentes dentro do campo, pela beleza de seus códigos.

Assim como o campo da arte, o campo *hacker* também está marcado pela dialética do refinamento, que segundo Bourdieu, "é o princípio do esforço que os artistas desenvolvem a fim de explorar e esgotar todas as possibilidades técnicas e estéticas de sua arte, em meio a uma pesquisa semi-experimental de renovação" (Boudieu – 2004:111). Dessa forma, o *hacker* tenta se afastar do programador de computadores convencional e se firmar como aquele que detém o conhecimento do fazer elegante.

Como foi dito anteriormente, caso o mantenedor de um *software* receba uma contribuição que não julgue pertinente ao escopo do projeto, pode simplesmente ignorar aquela contribuição. No entanto, é importante frisar que muitas contribuições são ignoradas por terem um código considerado "sujo", isto é, que percorrem caminhos muito longos para solucionar problemas muito pequenos e tornam o programa muito pesado, exigindo, assim, maior esforço do *hardware*. O que o mantenedor deve fazer é julgar a relação custo/benefício entre a implementação de uma solução de código sujo e a manutenção de um defeito do programa. Uma das coisas que mais se preza dentro do campo *hacker* é justamente esse "fazer elegante", isto é, a resolução de problemas complicados por meio de soluções simples. Em uma palestra em São Paulo, em 2004, organizada pela empresa 4linux, Jonh Mad Dog, da Linux International, tentou resumir o fazer elegante com a seguinte frase: "Ao invés de gastar milhões para desenvolver uma caneta capaz de escrever em gravidade zero, quando estiver nessas condições, o hacker usará um lápis.", ou seja, um código elegante não é um código rebuscado e sim, um código direto e eficiente.

Essa busca pela "beleza dos códigos" faz muitas vezes com que o campo passe a ignorar as demandas externas a ele, como as dos usuários finais e do mercado. Na prática, o que acaba acontecendo é que por mais que existam pressões por parte dos usuários, ou das empresas que se utilizam comercialmente de um *software* livre, as novas versões com as correções de possíveis problemas são lançadas de acordo com o ritmo de desenvolvimento de cada projeto e seguindo as regras de funcionamento da comunidade de *software* livre. No entanto, é importante frisar que o ato de não obedecer ao ritmo ditado pelo mercado para o lançamento de novas versões, nem sempre pode ser interpretado como lentidão na produção da comunidade, uma vez que a maioria dos *softwares* livres recebem atualizações em um ritmo muito maior dos que seus similares proprietários.

A explicação para o ritmo acelerado de atualização da comunidade vai além do

número de desenvolvedores e programadores envolvidos nos projetos, pois, quando uma empresa lança um software proprietário que é vendido, quase sempre a um custo relativamente alto, ela espera em média um ano até lançar uma nova versão, isso porque se é lançada a versão 1.0 de um programa e dois ou três meses depois é lançada a versão 2.0, os compradores da versão 1.0 se sentirão lesados, por serem obrigados a pagar em um espaço tão curto de tempo, para manter seus computadores atualizados, à medida que, os *softwares* livres podem ser adquiridos gratuitamente na Internet, juntamente com suas atualizações, as comunidades não precisam se preocupar com a insatisfação de seus usuários, quando estes recebem um comunicado de atualização, o que lhes permite lançar suas atualizações em espaços de tempo muito curtos.

Alguns dos parâmetros que podem ser utilizados para analisar o sucesso de um software livre são seu número de usuários e o de desenvolvedores que estão envolvidos no aperfeiçoamento do projeto. Contudo, o número de usuários não é o fator que promove maior distinção dentro da comunidade, isto porque o grau de especificidade de um software pode restringir muito seu número de usuários. Dessa forma, tal como no campo da arte, o campo *hacker* atribuirá maior distinção a um projeto, ou a um programador de acordo com as características técnicas de sua obra e o reconhecimento desta entre seus pares, que pode ser avaliado pelo número de envolvidos no desenvolvimento do projeto.

Como é possível notar, os produtos desenvolvidos pelos *hackers* são destinados a dois tipos de usuários: os leigos e os iniciados. Os leigos são os usuários finais, pessoas que se utilizam de ferramentas, como editores de texto, planilhas eletrônicas e outros aplicativos sem que tenham, ou que necessitem ter, algum conhecimento sobre programação, enquanto que os iniciados são outros *hackers,* ou pessoas com maior conhecimento técnico e com capacidade de interferir no desenvolvimento de uma solução, programando ou reportando erros aos mantenedores.

Esses dois públicos formam esferas de legitimação diferentes e com pesos

distintos dentro do campo. Se estivéssemos falando de uma empresa produtora de *softwares*, não haveria dúvida de que a esfera mais importante seria a dos leigos, pois estes formam o grande público consumidor de seus produtos. Contudo, quando falamos do campo *hacker,* esta não é a esfera mais importante, uma vez que os leigos não sabem o que acontece por tras das telas do modo gráfico; eles são incapazes de apreciar o código ou o algoritmo de um programa por mais genial que ele seja. Em outras palavras, os leigos são incapazes de reconhecer o trabalho de um *hacker*.

É evidente que criar ou participar de um projeto de grande aceitação do público acaba trazendo notoriedade para o desenvolvedor dentro do campo *hacker*. Contudo, durante as entrevistas que fiz, identifiquei entre estes intelectuais um discurso meritocrático, no qual se atribui maior ou menor distinção aos membros da comunidade de acordo com as suas contribuições para o Movimento de *Software* Livre, mais do que o seu sucesso de público.

Mas como funciona essedispositivo meritocrático de atribuição de distinção?

Quando um novo membro decide se juntar à comunidade, na maioria das vezes, ele se integra a um projeto que já está em andamento, atitude que é incentivada pela comunidade, que atualiza constantemente as listas de pacotes órfãos, isto é, pacotes que estão sem mantendores e que por isso não recebem atualizações com freqüência. À medida que o novo integrante for implementando melhorias no pacote que adotou ou de acordo com o número de contribuições aprovadas pelos mantenedores dos projetos dos quais faz parte, ele começa a ficar conhecido dentro das listas de discussão. Com isso sua fama cresce e outros *hackers* passam a admirar suas habilidades técnicas, e este novo membro da comunidade tem aumentadas suas chances de lançar, com sucesso, um projeto de sua autoria. Como a maioria dos projetos de *software* livre obedecem ao principio do "ditador benevolente", citado por Linus Torsvalds em sua obra *Só por prazer* (2001), para descrever a sua relação com os outros colaboradores no desenvolvimento do Linux ao aceitar ou recusar alguma colaboração, é importante

que o mantenedor do projeto tenha uma certa credibilidade junto aos outros desenvolvedores e que tenha uma capacidade técnica inquestionável, de forma que os últimos possam confiar nas suas decisões, quando ele aprova ou desaprova uma modificação no projeto.

Um princípio muito caro ao *software* livre e que aparece lado a lado com a discussão meritocrática, é o de que, tudo o que se pega da comunidade deve ser retribuído, pois apenas dessa forma ela pode continuar crescendo. Assim, a opinião de quem tem um número maior de contribuições à comunidade acaba tendo um peso maior na tomada de decisões, à medida em que ele tem maior capacidade de convencimento junto aos outros integrantes do projeto.

Como foi dito, um dos principais fatores de tomada decisão dentro do campo são as características técnicas de um projeto. No entanto, este não é o único fator relevante. É importante lembrar que toda a comunidade de *software* livre tem sua existência construída ao redor de um ou mais projetos. Estes projetos são estruturados sobre quatro pilares, isto é, as quatro liberdades que um *software* deve garantir a seus usuários para que possa ser considera do um *software* livre, a saber, a liberdade de executar o programa para qualquer fim, estudar seu funcionamento e modificá-lo para que possa atender melhor as necessidades do usuário, distribuir cópias do programa, e por fim, distribuir cópias alteradas do programa sem que seja necessário comunicar ao autor. Caso algum desses pilares seja quebrado, os projetos e as comunidades estruturadas em torno deles podem caminhar rapidamente para a extinção; por isso, foram criados alguns dispositivos de defesa dessas regras para garantir a sua perpetuação.

O principal mecanismo de defesa está nas licenças livres, cujo melhor exemplo é a GPL GNU – *General Public License* do projeto GNU (ANEXO 3). Quando um *software* é lançado, é necessário que se crie uma licença de uso para ele, isto é, um contrato firmado entre o produtor e os usuários, no qual estão expressas as regras de

uso e os termos de garantia daquele produto. Quando um *software* é lançado sob a GPL, o termo de uso garante ao usuário as quatro liberdades que são tão caras ao movimento e impõe uma regra de perpetuação dessas liberdades, que é a proibição de relicenciamento do *software,* ou de produtos derivados dele. Dessa forma, se uma pessoa adquire um *software* livre, não pode relicenciá-lo, mesmo que faça alterações significativas no código fonte do programa, nem pode licenciar um *software* derivado, isto é, que utiliza partes do código fonte de um programa licenciado sob a GPL com outra licença. Este tipo de licença é chamado por alguns de "licença viral", uma vez que contamina o *software* original e todos os produtos derivados que possam surgir a partir do seu código fonte ou de parte dele. (BOYLE -2005:48)

Este mecanismo impede por exemplo que empresas ou pessoas que não comunguem dos princípios da comunidade se apropriem do trabalho dos *hackers* e passem a ganhar dinheiro sobre esse trabalho sem retribuir à comunidade. Aqui é importante ressaltar que não existe, por parte do Movimento de *Software* Livre, a intenção de impedir o uso comercial de seus programas, muito pelo contrário. O que existe, é uma preocupação do movimento de garantir que o devido crédito seja atribuído aos autores dos programas e que quando uma empresa venha a utilizar um programa, ou partes de um programa cujo código é livre, ela não aprisione esse código e retribua à comunidade, devolvendo os códigos que produziu também sobre uma licença livre.

Quando analisamos a GPL, o mecanismo das licenças livres parece ser bem funcional para garantir a continuidade da liberdade dos códigos. Contudo, a GPL não e a única licença utilizada pelas comunidades. Existe uma série de outros modelos, alguns mais permissivos, outros menos. O autor de um programa pode escolher qual modelo de licença irá utilizar ou mesmo criar seu próprio modelo de licença, o que pode deixar brechas que permitam no futuro o aprisionamento dos códigos pelo autor, ou por terceiros. Este tipo de atitude é vista com extrema desconfiança pelo

movimento. Dessa forma, sempre que surge um novo projeto sob uma nova licença, esta é analisada pelos desenvolvedores interessados antes destes se integrarem ao projeto. Caso se identifique a intenção, ou mesmo a possibilidade daquele projeto se tornar um projeto proprietário, a comunidade reage quase sempre de forma virulenta contra a empreitada e grandes campanhas de desmoralização são iniciadas dentro das listas de discussão do movimento, a fim de esvaziar o desenvolvimento e desencorajar a implementação destes programas dentro de empresas ou qualquer instituição que possa legitimar o projeto.

Atitudes como essas reforçam o que diz Pierre Bourdieu a respeito da forma de organização de um campo de conhecimento, no qual,

... quanto mais o campo estiver em condições de funcionar como o campo de uma competição pela legitimidade cultural, tanto mais a produção pode e deve orientar-se para a busca das distinções culturalmente pertinentes em um determinado estágio de um dado campo, isto é, busca dos temas, técnicas e estilos que são dotados de valor na economia específica do campo por serem capazes de fazer existir culturalmente os grupos que os produzem, vale dizer, de conferir lhes um valor propriamente cultural atribuindo-lhes marcas de distinção (uma especialidade, uma maneira ou um estilo) reconhecidas pelo campo como culturalmente pertinentes e, portanto, suscetíveis de serem percebidas enquanto tais, em função das taxinomias culturais disponíveis em um determinado estágio de um dado campo. (BOURDIEU – 2004 : 109)

Dessa forma, ao mesmo tempo em que o campo estimula a disputa pela distinção, ele impõe claramente as regras que devem ser seguidas nesta disputa e rechaça qualquer tentativa de distinção não legítima que possa pôr em risco a existência do campo.

Estas características transformam o campo em uma arena na qual é disputada a

legitimidade tecnológica e a produção de bens tecnologicamente pertinentes ao Movimento de *Software* Livre. Estão em competição constante distribuições, projetos e desenvolvedores que buscam sua legitimidade dentro do movimento. Essa mescla de elementos competitivos e colaborativos constitui a base para o modelo econômico que sustenta a comunidade de *software* livre. Dessa forma, como afirma Gurovitz:

Programas individuais podem competir dentro do ambiente Linux e empresas individuais também podem competir para aperfeiçoar o sistema e vender a melhor distribuição ou a mais fácil de usar e instalar, mas, já que ninguém é dono do software o esforço coletivo constrói um corpo cooperativo único, temos então, a maior produtividade e eficiência de um ambiente competitivo livre com um resultado tecnicamente melhor e mais coerente de um produto que passa por uma revisão dos maiores especialista. (GUROVITZ – 2002 : 69, 70)

um questionamento que surge com freqüência é, como isso pode ser economicamente viável, por que uma pessoa compraria um *software* que ela pode baixar gratuitamente da Internet?

O modelo econômico proposto pelo Movimento do *Software* Livre não se baseia na venda de seus *softwares*, nem acha que os bens mais valiosos sejam as possíveis patentes de seus códigos fonte. Ele se baseia na venda de serviços. Robert Young, um dos fundadores da Red-Hat – empresa que mantém uma das mais populares distribuições GNU/LINUX do mercado – traça um paralelo entre a Movimento de *Software* Livre e as oficinas mecânicas que podem ser encontradas em todas as cidades do mundo; segundo ele, mesmo que as pessoas pudessem comprar separadamente todas as peças de seu carro, poucos teriam vontade, ou conhecimento técnico suficiente para montá-lo no quintal de casa. Então, compram Chevrolet, Chrysler, Honda ou Toyota e sempre que têm um problema com o carro, procuram um mecânico em vez de tentar concertá-lo sozinhas. (GUROVITZ – 2002 :70)

Ao distribuir gratuitamente, ou a um custo muito baixo, o sistema operacional e seus aplicativos, acompanhados de seus códigos fonte, a comunidade cresce a uma velocidade assombrosa. Os custos que beiram a zero atraem cada vez mais novos usuários, o que cria a demanda do serviço desses mecânicos de *software*, que são atraídos pelo mercado de trabalho e pela possibilidade de acesso ao código fonte dos programas, que lhes permite crescer técnica e profissionalmente.

Tal como o campo de produção erudita, o campo de produção *hacker* também estabelece relações com o campo das instâncias de difusão do conhecimento técnico específico produzido por estes intelectuais e com as instâncias de reprodução dos saberes exigidos para se adquirir e decifrar esse conhecimento, sendo que todas essas instituições disputam o que Bourdieu chama de "monopólio legítimo do uso do poder simbólico". (BOURDIEU – 2004 : 118)

O campo de produção *hacker* leva mais em conta a contribuição de cada um desses intelectuais no fortalecimento institucional do movimento do que seus méritos acadêmicos. No entanto, a academia exerce um papel importante ao formar o público iniciado do qual falei anteriormente, pessoas que não necessariamente se tornam *hackers*, mas que possuem o conhecimento técnico necessário para entender e admirar o trabalho desses intelectuais. Justamente por terem estes conhecimentos, estes iniciados podem influir no desenvolvimento de certos aplicativos, não só programando, mas reportando possíveis erros. Por fim, acabam se transformando em uma importante esfera de legitimação à medida que implementam esses aplicativos em seus locais de trabalho, reconhecendo a qualidade desse modelo de desenvolvimento.

À proporção que as grandes empresas de *hardware* e *software* começaram a se interessar por este tipo de tecnologia, e principalmente por este modelo de desenvolvimento, novas instituições passaram a pleitear a legitimidade para certificar, não só projetos, como também profissionais da área de *software* livre. O exemplo de maior expressão no mundo dessas agências certificadoras é a LPI (*Linux Professional*

Institute), que é uma organização não-governamental (ONG) sem fins lucrativos, mantida por várias companhias de renome mundial, tais como IBM, Novell, SGI, SuSE Linux, HP, MandrakeSoft, Intel, LinuxJournal entre outras, que tem como missão principal criar e manter programas de certificação profissional em sistemas GNU/Linux, que atestem os conhecimentos de profissionais em diversas áreas deste sistema operacional.[2]

Essas instituições surgem para atender uma necessidade exclusiva do mercado e de alguns profissionais perante este mercado, uma vez que nem todos os projetos de um *hacker* de destaque no campo se tornam um sucesso e nem todos os desenvolvedores atrelados a um projeto de sucesso recebem o mesmo grau de reconhecimento, uma vez que este deriva da quantidade e qualidade das contribuições de cada um. Começam a surgir, então, algumas "instâncias de legitimação e consagração" (BOURDIEU – 2004) voltadas para os desenvolvedores, outras para os projetos e, por conseqüência, para os desenvolvedores de maior destaque dentro desses projetos.

Contudo, os membros do campo acabam enxergando com desconfiança certas instituições que pleiteiam esse papel de certificadora. Se, por um lado, o mercado vê com bons olhos um profissional que possua um diploma de ciência da computação e um certificado LPI, o campo *hacker* ignora completamente este profissional, caso ele não tenha feito contribuições relevantes à comunidade. Justamente por estas instâncias estarem, de alguma forma, representando os interesses de instituições privadas, muitos membros da comunidade não reconhecem nessas agências a legitimidade necessária para que certifiquem que um membro do campo tem conhecimentos suficientes sobre um sistema operacional que ele mesmo ajudou a criar.

Depois dos projetos propriamente ditos, as maiores instituições de legitimação dentro do campo *hacker* são as distribuições, que acumulam também o papel de

2 para mais informações sobre a LPI, ver www.lpi.org.br

difusoras dessa produção, sendo que cada uma delas também atribui um grau diferenciado de distinção aos seus desenvolvedores, tradutores e empacotadores, que são respectivamente os responsáveis por desenvolver *softwares* específicos para a distribuição, traduzir os *softwares* que a compõem para sua língua de origem e, por fim, os responsáveis por prospectar, testar e configurar cada um dos *softwares* que irão compor a distribuição.

Atualmente, existe um incontável número de distribuições, mas, em geral, o que se pode notar é que as distribuições que são mantidas por empresas, acabam por atribuir um grau menor de distinção a seus usuários e desenvolvedores do que as distribuições que são mantidas exclusivamente com esforços da comunidade, como é o caso do Slakware ou do Debian.

Por ter criado uma hierarquia muito bem definida e um rígido sistema de admissão de novos desenvolvedores, controle de versões, votação e tomadas de decisões, o Debian é hoje uma das distribuições que atribuem maior grau de distinção aos poucos *hackers* que conseguem o título de desenvolvedor oficial Debian.

Como existe um controle muito rígido de todos os *softwares* que compõem a versão estável do Debian, quando um programa é escolhido para se tornar, por exemplo, o navegador de Internet oficial, ou mesmo a sua interface gráfica padrão, isso acaba dando uma enorme visibilidade a esses projetos, bem como aos seus desenvolvedores, mesmo que eles não estejam ligados diretamente ao Debian, como desenvolvedores oficiais da distribuição. Por este motivos, o modo de trabalho do Debian passou a ser adotado como modelo por inúmeras comunidades de *software* livre e transposto para várias ações menores, inclusive em projetos que não são relacionados especificamente ao desenvolvimento de *softwares*, como é o exemplo da publicação mensal DebianZine, desenvolvida pela comunidade de usuários Debian do Brasil.[3]

3 O DebianZine pode ser acessado pelo do endereço http://cdd.debian-br.org

Como disse anteriormente, o objetivo desse capítulo é contextualizar historicamente o leitor e lhe fornecer alguns conceitos básicos para que possa entender o funcionamento do Movimento de *Software* Livre. No próximo capítulo, no entanto, será retomada a discussão sobre o funcionamento das distribuições, mais especificamente da distribuição Debian e como os conceitos descritos nas páginas anteriores se aplicam a este caso específico.

9 Capítulo – 2

10 A organização da comunidade Debian

O Movimento de *Software* Livre é composto por várias comunidades, algumas maiores outras menores, cada uma com características próprias. Geralmente as comunidades se formam em torno de um projeto, ou de uma distribuição. Este tipo de divisão permite que existam ao mesmo tempo comunidades com características muito específicas e tamanhos diversos, como é o caso do grupo de desenvolvedores responsável pela criação, manutenção e aprimoramento do Openoffice, um pacote com programas de escritório para a plataforma GNU/Linux e a comunidade responsável pelo desenvolvimento da distribuição Debian que agrega o pacote Openoffice dentro de sua solução.

As diversas distribuições do GNU/Linux que se encontram em atividade também apresentam características muito distintas, dependendo do público que elas pretendem atingir. Hoje é possível encontrar distribuições voltadas para o usuário doméstico, uso corporativo, para a produção multimídia, servidores de Internet, ambientes de desenvolvimento, computadores de grande porte e muito mais.

Compostas pelo conjunto de *softwares* que os seus organizadores acham necessários para que um computador funcione a contento, seja ele um computador pessoal, uma máquina corporativa ou uma estação multimídia, as distribuições são hoje uma das principais esferas de consagração e legitimação do trabalho dos *hackers*. Quando comunidades em torno de distribuições de grande projeção no Movimento de *Software* Livre como a Debian, Slakware, SUSI, ou Red-Hat, adotam um navegador de Internet ou uma opção de ambiente gráfico, para serem os *softwares* padrões da sua distribuição é como se estes grupos estivessem dizendo, “estes *softwares* são tão eficientes quanto a nossa distribuição”. Esta atitude, que implica em uma consagração da relevância tecnológica do programa de computador escolhido, também se estende

aos técnicos envolvidos no desenvolvimento do *software* em questão, atribuindo-lhes maior ou menor distinção dentro da comunidade.

Da mesma forma que Bourdieu afirma que:

não se pode compreender inteiramente o funcionamento e as funções sociais do campo de produção erudita sem analisar as relações que mantém, de um lado, com as instâncias, os museus por exemplo, que tem a seu cargo a conservação do capital de bens simbólicos legados pelos produtores do passado e consagrados pelo fato de sua conservação e, de outro lado, com as instâncias qualificadas, como por exemplo o sistema de ensino, para a assegurar a reprodução do sistema de esquemas de ação, de expressão, de concepção, de imaginação, de percepção e de apreciação objetivamente disponíveis em uma determinada formação social. (BOURDIEU 2004:117)

Também não é possível pensar o campo de produção *hacker* sem levar em conta a relação que ele mantém com as instituições certificadoras, que atestam o conhecimento técnico desses profissionais, com as distribuições que atestam a qualidade dos produtos desenvolvidos por eles e com a academia que forma, não só os *hackers,* mas o público qualificado, capaz de entender o trabalho desses intelectuais.

Tanto as distribuições, como as agências certificadoras e a academia, competem entre si por uma distinção tecnológica e pelo poder de concedê-la. Contudo, como trabalham junto a públicos diferentes, elas concedem graus diferentes de legitimação dentro do campo, isto porque um profissional com um título de analista de sistemas, cedido por uma universidade, ou com uma certificação da LPI, que ateste seus conhecimentos sobre o sistema operacional GNU/Linux, pode ou não se tornar um *hacker.* Na realidade, este rótulo só pode ser concedido por outros *hackers,* que levarão em conta a contribuição que o profissional em questão fez ao movimento, ou a uma comunidade específica.

Os títulos concedidos pela academia ou pelas agências certificadoras têm maior influência no mercado de trabalho do que no campo *hacker* propriamente dito. As empresas querem uma garantia de que esses profissionais sabem fazer o que eles dizem

que fazem e essas agências certificam exclusivamente este conhecimento. Assim, da mesma forma que a arte precisou se desvencilhar das esferas de legitimação externas, liberando-se não só economicamente, mas também das demandas éticas e estéticas da igreja e da aristocracia, os *hackers,* quando surgem como um movimento organizado, também precisam se desvencilhar das demandas éticas e estéticas da indústria do *software,* para que possam se firmar como um campo autônomo de produção de conhecimento, o que faz com que as agências certificadoras e a academia transmitam um grau muito menor de legitimidade do que as comunidades, sejam elas organizadas em torno de um projeto específico, ou de uma distribuição.

Esse desejo de rompimento tem como principal marco o momento em que os *hackers* fazem a opção pelo licenciamento permissivo de seus programas, utilizando licenças livres como a GPL. Contrariando a lógica do mercado, eles dizem "todos podem copiar, vender, alterar e distribuir livremente o nosso trabalho". A partir desse momento, os *hackers* param de receber pelas cópias vendidas de seus programas e passam a receber por serviços prestados, como a adaptação ou desenvolvimento de *softwares* para tarefas específicas, a exemplo do gerenciamento de uma padaria.

Para trabalharem sob essa nova ética, o trabalho desses intelectuais passa a se organizar de forma colaborativa. *Hackers* com propósitos e projetos similares passam a se organizar em comunidades que recuperam o que Thorstein Veblem chamou de "instinto do trabalho bem feito".

"instinto de trabalho bem feito" foi o termo que Veblen escolheu para um "gosto natural pelo trabalho efetivo e um desapreço pelo esforço "fútil", em sua opinião, presente em todos os homens.[..] Se todos nos orgulhamos de um trabalho bem feito, também temos, é o que sugere Veblem, uma repulsa pela labuta sem propósito, pelo esforço fútil, pela azáfama sem sentido. (BAUMAN – 2003 : 31)

Em sua obra intitulada *Comunidade – A busca por segurança no mundo atual* (2003) o sociólogo Zygmunt Bauman procura mostrar que, ao remover o homem da intrincada teia de relações que dotavam o seu trabalho de sentido e colocá-lo no frio

chão da fábrica, a Revolução Industrial deu início a um processo lento e gradual de destruição das comunidades tradicionais, pondo fim à possibilidade de realização do instinto do trabalho bem feito. Na verdade, a teia de relações comunitárias, era o que estabelecia a diferença, dotava o trabalho de sentido, fazendo do mero empenho uma atividade significativa, uma ação com objetivo. Era esta teia de relações, nas palavras de Bauman, que constituía a diferença entre o "esforço", ligado aos conceitos de dignidade, mérito e honra, e a "labuta", não ligada a qualquer desses valores e portanto percebida como fútil.

Segundo Bauman, depois da Revolução Industrial, o que costumava ser visto como um "esforço", nos termos de Veblem, transformou-se em "labuta".

Já não era claro para os artífices e artesãos de ontem o sentido do "trabalho bem feito", e não havia mais "dignidade, mérito e honra" que decorressem dele. Seguir a rotina sem alma do chão da fábrica, sem ser observado pelo companheiro ou pelo vizinho, mas apenas pelo desconfiado capataz, obedecer aos movimentos ditados pela máquina sem chance de admirar o produto do próprio esforço, e muito menos de apreciar a sua qualidade, tornavam o esforço "fútil". (BAUMAN - 2003:32)

No início do Movimento de *Software* Livre, os *hackers* retomam o hábito de observar e serem observados pelos seus companheiros, consolidando o caráter comunitário de suas ações, e ao mesmo tempo, contribuindo para fortalecerem-se em um campo autônomo de produção de conhecimento. Os trabalhos realizados para a comunidade deixaram de ser tidos apenas como "labuta" e passam ser interpretados como "esforço" em benefício de todos. Associando novamente o trabalho aos conceitos de honra, dignidade e principalmente de mérito, os esforços realizados dentro da comunidade concedem aos *hackers* a admiração de seus companheiros.

A busca pelo reconhecimento entre seus pares, isto é, os profissionais que dominam as mesmas técnicas e linguagens específicas do campo, faz com que os *hackers* tentem levar essas técnicas a seu limite, refinando ao máximo os seus códigos antes de lançá-los para a apreciação da comunidade.

A partir do momento em que o programador passa a ter os códigos fontes de seus *softwares* analisados por outros programadores, que poderão sugerir novas funcionalidades, criticar ou se juntar ao autor original para colaborar com o aprimoramento do *software*, ele passa a se preocupar, não só com a funcionalidade de seu programa, mas também com a "beleza" dos códigos que estão por trás de seu funcionamento, pois este será o principal diferencial entre ele e os outros programadores. Mediante seus códigos e a relevância deles para o movimento, ou para a comunidade, o *hacker* alcançará maior, ou menor distinção dentro do campo.

Todavia, como poderíamos definir o que é um código "belo", como atribuir beleza a uma seqüência lógica de comandos em um arquivo de texto?

Segundo Fernando Ike[4], um dos colaboradores do projeto Debian no Brasil, qualquer argumentação que se foque apenas na "beleza" do código, levando em conta apenas o caráter estético, é muito fraca na prática, porque um código bonito ou elegante pouco importa se ele não funciona, ou não serve para coisa alguma. Assim, a "beleza" do código de um desenvolvedor pode ser vista em sua clareza e funcionalidade. A função de uma linguagem de programação é a mesma de um idioma como o português, ou o inglês, o principal motivo de sua existência é a comunicação. A linguagem de programação consiste em uma forma de comunicação entre o ser humano e o computador.

Na oportunidade em que o entrevistei, Fernando Ike apresentou o seguinte exemplo para a definição de um código "belo":

> *Imagine que um poema muito grande e rebuscado, que é considerado uma obra prima por alguns especialistas, pode ser considerado uma porcaria por um cara que estuda no supletivo noturno na periferia da cidade, simplesmente porque ele não o entende. Contudo, ele pode considerar como obra prima uma letra de Rap, porque ele a entende.*
>
> *Agora para completar, imagine uma letra de música popular brasileira,*

4 Entrevista concedida em agosto de 2006

> *ela lida como poema pode ser admirada por um especialista em Letras ou por um fã de Rap, pois é de fácil compreensão, já que as palavras usadas são comuns a ambos. O mesmo acontece com as linguagens de programação, o mais importante é escrever um código funcional e que o maior número de pessoas possa entendê-lo.*

Assim, a beleza de um código está justamente em sua simplicidade e funcionalidade. Se um *software* apresenta um alto grau de funcionalidade e possui um código relativamente simples e claro, a probabilidade de o autor original conseguir formar uma comunidade em torno do seu projeto é muito maior. Quanto mais claro for o código e quanto maior o número de comentários que expliquem trechos mais complicados do código ou determinadas escolhas técnicas feitas pelo autor original, mais fácil é tal *software* ser analisado pela comunidade, que pode ajudar a solucionar problemas, sugerir novas funcionalidades ou questionar determinadas decisões dos mantenedores do projeto.

Nas comunidades tradicionais ao contrário das sociedades modernas, segundo Ferdinand Tönies, o entendimento entre os seus membros se dá de forma tácita.

> *O entendimento ao estilo comunitário, casual (zuhanden, como diria Martin Heidegeer), não precisa ser procurado, e muito menos construído: este entendimento já "está lá" , completo e pronto para ser usado – de tal modo que nos entendemos "sem palavras" e nunca precisamos perguntar com apreensão, "o que você quer dizer?". O tipo de conhecimento em que a comunidade se baseia precede todos os acordos e desacordos. Tal entendimento não é o ponto de chegada mas o ponto de partida de toda a união.* (TÖNIES 1963 apud BAUMAN 2003:15).

Ao contrário dessas comunidades tradicionais, que segundo Ferdinand Tönies (*idem*), encontravam sua coesão a partir de um entendimento compartilhado por todos

os seus membros, as comunidades de *software* livre, buscam a coesão a partir do consenso em torno de decisões técnicas tomadas pelos mantenedores dos projetos.

Assim como todas as comunidades de *software* livre se organizam em torno de um projeto, todo projeto é coordenado por um mantenedor, ou um grupo de mantenedores, que por algum motivo se destacam dos outros membros da comunidade, seja por terem sido os fundadores do projeto, seja por suas contribuições a ele. O fato é que, são os mantenedores do projeto que decidem quais contribuições serão aceitas e quais modificações serão implementadas na próxima versão do *software,* ou da distribuição que estiver em questão.

É claro que tais decisões são tomadas somente depois de muitos debates entre os membros da comunidade a fim de se tomar a decisão que pareça mais adequada a todos. No entanto, é necessário lembrarmos que a construção do entendimento a partir do consenso, utilizado pelas comunidades de s*oftware* livre é um acordo, muitas vezes, alcançado entre pessoas de opiniões essencialmente diferentes, isto é, um produto de negociações e compromissos difíceis, o que pode envolver muita disputa e contrariedades entre as partes envolvidas. Por isso, essas comunidades têm estatutos e regras de comportamento muito bem definidas para que possam assegurar a coesão do grupo.

Ao contrário das comunidades tradicionais, nas quais o entendimento compartilhado era do tipo tácito e passava desapercebido por seus membros, nas comunidades de s*oftware* livre, a construção do entendimento se dá de forma explícita e, por isso, precisa ser lembrada a todo momento, como forma de internalizar estas regras nos indivíduos que as compõem. Segundo Bauman (2003), esse comportamento se explica porque uma negociação prolongada também pode resultar em um acordo que, se obedecido diariamente, pode, por sua vez, tornar-se um hábito que não precisa mais ser repensado, e muito menos monitorado ou controlado.

A comunidade formada em torno da distribuição Debian é um grande exemplo, não só dessas comunidades construídas explicitamente, isto é, a partir do consenso, mas de como as regras claras nas tomadas de decisão e o cuidado com a qualidade técnica de seus produtos e de seus desenvolvedores e colaboradores pode se tornar um forte instrumento de distinção de seus membros junto ao restante do Movimento de *Software* Livre.

Embora estes *hackers* se auto-denominem membros de uma "comunidade", suas relações são permeadas por disputas que os aproximam mais do que Bourdieu denominou como "campo". Como vimos anteriormente, as relações que se estabelecem entre os desenvolvedores envolvidos em um projeto, não se dão de forma horizontal, elas são regidas pelo nível de distinção técnica de cada um dentro da "comunidade".

As distribuições são hoje as maiores instâncias de legitimação desse campo *hacker;* para que esta relação possa ficar mais clara, tomaremos com exemplo o funcionamento formal da comunidade Debian e como se da o processo de eleição de suas esferas de decisão, onde é possível ver essa ambiguidade entre o conceito de campo, propostos por Bourdieu e a visão idealizada de "comunidade", exaltada no discurso dos militantes do Movimento de *Software* Livre.

Um dos motivos da escolha dessa comunidade em detrimento de outras tantas se deu justamente por sua organização e a forma como enfrenta os conflitos internos com os quais se depara. Durante a escolha, pesou também o grau de influência que ela exerce sobre todo o Movimento de *Software* Livre e o fato da distribuição ser mantida exclusivamente com os esforços de programadores voluntários, e não por uma empresa ou universidade em particular. As regras de comportamento do Debian, e de seus membros, não são dadas por interditos, nem de forma tácita, estão expressas de maneira clara e objetiva em seu contrato social (ANEXO – 2); lá estão os compromissos do projeto e de seus desenvolvedores junto à comunidade de *software*

livre e seus usuários, bem como a definição de *software* livre adotada pelo projeto e as licenças aceitas pelo Debian.

Criado em 16 de agosto de 1993, por Ian Murdock, o projeto Debian tinha como objetivo propiciar uma distribuição GNU/Linux 100% livre, o que nenhuma outra distribuição havia conseguido até então.

A idéia de declarar seu "contrato social para a comunidade de software livre" foi sugerida por Ean Schuessler. O rascunho deste documento foi escrito por Bruce Perens e refinado por outros desenvolvedores Debian durante uma conferência via e-mail que durou um mês, em junho de 1997, e só então foi aceita como uma política pública do Projeto Debian. Mais tarde, Bruce Perens removeu as referências específicas do Debian da Definição Debian de *Software* Livre para criar a Definição de Código Aberto. (DEBIAN - 1997)

De novembro de 1994 a novembro de 1995, o projeto Debian foi patrocinado pela *Free Software Foundation* (FSF). No entanto, o Debian era uma distribuição GNU/Linux, isto é, sistema GNU com *kernel* Linux, e a FSF não havia abandonado seu projeto de fazer um sistema inteiramente GNU, que pretendia trabalhar com um sistema de *microkernels*, chamado HURD.

Em novembro de 1995, a FSF retirou seu patrocínio, o que forçou Bruce Perens, o então líder do projeto Debian, a concentrar seus esforços na criação da *Software in the Public Interest* (SPI)[5] – Software de Interesse Público – uma organização sem fins lucrativos que nasceu com o objetivo de arrecadar e gerir recursos para o projeto Debian. Rapidamente, a SPI passou a ajudar, não só o Debian, como também outros projetos de *software* livre e de *hardware* livre. Até o momento em que esta pesquisa estava sendo fechada, em janeiro de 2007, além do projeto Debian a SPI recebia e geria recursos para os projetos Freco, GNUstep, OFTC, PostgreSQL, Open Source, GNU

5 Para mais informações sobre a SPI, consultar o site www.spi.org

TeXmacs, wxWidgets, Drupal, OpenVAS e a Open Voting Foundation.

À medida que o Debian foi crescendo, a sistematização da forma como os colaboradores do projeto deveriam trabalhar também foi evoluindo. No momento em que o Debian foi lançado, já existiam outras distribuições Linux disponíveis no mercado; porém como ressaltava Ian Murdock no Manifesto Debian (ANEXO - 4), quando se referia às distribuições:

[...]elas não são simples nem 'legais' de construir e requerem uma grande quantidade de esforço e tempo de seus criadores para que elas mantenham-se livres de erros e sempre atualizadas. Uma coisa é criar um sistema do 'nada'. Outra coisa é ter certeza que o sistema é fácil dos outros instalarem, que funcionará com uma larga variedade de configurações de hardware, que conterá programas que serão úteis aos outros, e que será atualizado quando seus componentes são melhorados.(MURDOCK - 1994)

O que ocorria com a maioria das distribuições na época é que estas eram mal organizadas, seus programas não funcionavam direito, e acabavam gerando uma frustração nos desenvolvedores que tinham seu primeiro contato com o GNU/Linux. Além do que, essas distribuições tidas como comerciais, freqüentemente omitiam o fato do Linux ser um *software* livre e estar disponível sobre a licença GPL.

Quando o projeto Debian foi criado, ele buscou enfrentar esses problemas da mesma forma que Linus Torvalds (2001) fez com o Linux, isto é, abrindo o desenvolvimento da distribuição para o maior número de pessoas possível, pois dessa forma, muitos poderiam contribuir com sugestões, localização e correção de erros, além de poder alocar as pessoas nas áreas, nas quais tinham maior interesse e experiência.

Para que possamos entender melhor como se dão os processos de colaboração e tomadas de decisão dentro comunidade Debian, vamos dividí-la em três blocos. O

primeiro bloco é o dos desenvolvedores oficiais do projeto, o segundo bloco é o dos colaboradores que podem vir, ou não, a se tornar desenvolvedores oficiais do projeto e o terceiro bloco é o que é formado pelos usuários finais, isto é, aqueles que utilizam a distribuição Debian, mas não necessariamente colaboram para o desenvolvimento do projeto, ou têm os conhecimentos necessários para fazê-lo.

Os membros da comunidade que trabalham como colaboradores tomam decisões exclusivamente sobre seu trabalho individual, ou sobre os pacotes que mantêm, se este for o caso. Todas as tomadas de decisão sobre a política do projeto são feitas exclusivamente pelo primeiro bloco, isto é, o dos desenvolvedores oficiais, que para isso se organizam dentro das seguintes instâncias decisórias:

1. o líder do projeto;
2. o secretário do projeto
3. o comitê técnico
4. os delegados apontados pelo líder do projeto para tarefas específicas;
5. o desenvolvedor individual e;
6. o conjunto dos desenvolvedores (que pode ser interpretado como uma assembléia geral).

Estrutura para a tomada de decisões

Anualmente a comunidade Debian organiza uma votação para a escolha do líder do projeto; o processo de escolha acontece durante as nove semanas que precedem o final do mandato que estiver em vigor. A votação para a escolha do líder, assim como todas as outras votações que são realizadas dentro do Debian, é coordenada pelo secretário do projeto, que recebe, durante as três primeiras semanas, as inscrições de todos os desenvolvedores que se interessam em concorrer ao cargo. Todos os desenvolvedores podem pleitear o cargo de líder; os candidatos devem enviar, junto com seu pedido de inscrição, uma plataforma com suas propostas para o projeto. O objetivo é que todos os desenvolvedores possam enviar perguntas e sugestões aos candidatos. Durante as três últimas semanas, é realizada a votação e escolhido o novo líder.

O líder do projeto deve representar o Debian junto às outras instituições e empresas que atuam dentro do Movimento do *Software* Livre, tentando firmar parcerias e manter boas relações com estes agentes. Internamente, o líder deve mediar as relações entre os desenvolvedores e entre as outras esferas de decisão do Debian. Entre as atribuições desse cargo estão a de nomear delegados ou delegar decisões ao comitê técnico. Dessa forma, quando é solicitado a tomar alguma decisão sobre a qual não tenha pleno domínio, o líder pode nomear um delegado para cuidar especificamente daquela questão, pedir aconselhamento, ou delegar a decisão ao comitê técnico.

Ele deve interagir com os desenvolvedores a fim de ajudar na solução de disputas que possam vir a ocorrer, tentando buscar um consenso entre as partes envolvidas, podendo dar suporte a pontos de vista, ou a determinados desenvolvedores do projeto. Além disso, é dele o dever de tomar todas as decisões que necessitem de urgência, ou

que não estejam previstas nas atribuições de nenhuma das instâncias decisórias do Debian.

Em conjunto com a *Software in the Public Interest* (SPI), o líder do projeto Debian toma as decisões que afetam as propriedades guardadas em confiança para propósitos relacionados ao projeto, isto é, cabe a ele tomar as decisões sobre o uso do dinheiro e dos equipamentos que são doados ao Debian.

Qualquer pessoa pode acumular vários cargos dentro do projeto Debian exceto o secretário do projeto, que não pode acumular os cargos de líder nem de diretor do comitê técnico. Ao contrário dos outros delegados que são nomeados pelo líder, o futuro secretário, além dessa indicação deve ter a aprovação do atual secretário do projeto. Caso não seja possível um consenso entre o líder e o secretário atual, a mesa da SPI é convocada para indicar um novo secretário.

O mandato do secretário é de um ano, podendo ser reencaminhado ao cargo. Entre suas atribuições, estão a condução das votações do projeto, a ocupação de cargos em vacância e a interpretação da Constituição Debian.

Além das votações para líder do projeto, o secretário deverá conduzir todas as votações que ocorrerem dentro da comunidade, determinando o número e a identidade das pessoas elegíveis para que seja possível calcular o quórum da votação.

O secretário do projeto, junto com o diretor do comitê técnico podem substituir o líder e tomar decisões em seu lugar. Este recurso é utilizado apenas quando é extremamente necessário, isto é, quando o cargo em questão estiver em vacância e os substitutos tiverem o apoio dos demais desenvolvedores. O secretário também é responsável por arbitrar qualquer disputa de interpretação da constituição Debian.

Formado por no mínimo quatro (4) e, no máximo, oito (8) membros, o comitê técnico é responsável por decidir sobre as políticas técnicas do projeto, como por

exemplo os manuais e materiais de referência para os desenvolvedores.

O comitê pode decidir sobre qualquer assunto técnico no qual haja a sobreposição de jurisdição dos desenvolvedores. Por exemplo, no caso de os desenvolvedores de um pacote não concordarem com as prioridades, ou o curso que deve ser tomado dentro de um projeto, eles podem encaminhar o problema para que o comitê tome a decisão. O comitê técnico só toma uma decisão como última alternativa.

Caso os desenvolvedores de um pacote não consigam chegar a um consenso sobre qual caminho tomar, devem elaborar um texto, de comum acordo, explicando a situação e encaminhá-lo para o comitê técnico. O assunto será discutido em uma lista de *e-mails* aberta, a qual todos os interessados podem ter acesso. Durante a discussão, é possível que uma das partes seja convencida, então, ela comunica sua satisfação com os argumentos apresentados à lista e não é necessário que o comitê tome uma decisão formal; caso nenhuma das partes se de por convencida, esta instância tentará tomar a decisão mais adequada possível. Além dessas situações, qualquer pessoa ou instância decisória, pode delegar uma decisão ao comitê técnico ou buscar aconselhamento junto a ele.

O comitê também pode sobrepujar um desenvolvedor, pedindo para que ele tome um determinado curso no desenvolvimento de seu pacote, mesmo que o último não concorde com com a posição do órgão. Uma ação desse tipo requer maioria de três (3) para um (1) no comitê e pode acontecer quando esta instância decisória achar que a reclamação de um emissor de erro é justificada e que a solução proposta por ele deve ser implementada. Nesse caso, não existe nenhum mecanismo que obrigue o desenvolvedor a seguir tais instruções, mas ele não deve, de forma alguma, trabalhar de forma a impedir que estas resoluções sejam implantadas.

Os delegados que são apontados pelo líder do projeto têm poder de decisão

apenas sobre os assuntos específicos para os quais foram nomeados e para assuntos sobre os quais o líder não pode decidir diretamente, como por exemplo, a aprovação ou remoção de desenvolvedores no projeto. Este mecanismo serve, em particular, para evitar a concentração de poder nas mãos do líder, principalmente sobre a entrada de novos desenvolvedores no projeto. O líder pode, a qualquer tempo, nomear ou destituir um delegado, mas não pode reverter nenhuma decisão tomada por eles.

Os desenvolvedores são voluntários que ajudam no aperfeiçoamento do projeto Debian, mantendo pacotes, ou realizando algum trabalho que seja considerado importante pelos delegados do líder do projeto. O desenvolvedor individual tem autonomia para tomar qualquer decisão sobre seu próprio trabalho, sendo ela técnica ou não. Todo desenvolvedor pode votar e se candidatar ao cargo de líder do projeto. É possível ainda, que ele pode vote, apadrinhe ou proponha rascunhos de resoluções gerais, que são votadas pelo conjunto dos desenvolvedores.

Uma resolução geral aprovada pelo conjunto dos desenvolvedores, tem o poder de modificar a constituição Debian, desde que tenha maioria de três (3) para um (1), anular qualquer decisão tomada pelo líder, ou por um de seus delegados; anular qualquer decisão tomada pelo comitê técnico, desde que tenha maioria de dois (2) para um (1); Criar, substituir ou retirar documentos e declarações de políticas não técnicas.

Para que uma resolução geral seja posta em votação, é necessário que um desenvolvedor apresente um rascunho e que sua proposta seja apadrinhada por K desenvolvedores. Tais resoluções devem ter um quórum de pelo menos 3Q desenvolvedores. Para calcular as variáveis K e Q, o secretário do projeto leva em conta as seguintes fórmulas; Q é igual à metade da raiz quadrada do número de desenvolvedores do projeto; se Q for menor ou igual a 5, então K é igual a Q, se Q for maior que 5 então K é igual a 5. Em 2004, na eleição para o líder do projeto, o Debian contava com novecentos e onze (911) desenvolvedores oficiais espalhados pelo mundo. Dessa forma, para se colocar em votação uma resolução geral em 2004, era necessário

que a proposta fosse apadrinhada por cinco (5) desenvolvedores, e a votação deveria ter o quórum mínimo de 45,2741648183597 desenvolvedores. Devemos observar que a constituição Debian especifica que as variáveis Q e K não precisam ser números inteiros e não devem ser arredondados.

É importante lembrar que a todo momento a comunidade Debian busca o consenso entre seus membros. Dessa forma, uma questão só é levada à votação em última instância, depois que todas as possibilidades de convencimento das partes envolvidas se esgotaram por meio de longos debates. Em geral, uma das partes é convencida durante o processo de discussão, e a proposta de votação é retirada.

Tanto na eleição do líder do projeto, como na votação de suas resoluções gerais, a comunidade busca a construção de um consenso. Para isso, é utilizada nas votações uma variação do método de Condorcet, chamada jogo de Schwartz. Como a construção do consenso quase sempre é traumática, já que significa o acordo entre partes com pontos de vistas diferentes, que acabam abrindo mão de valores que consideram importantes para garantir uma situação aceitável para todos os envolvidos, a comunidade Debian busca escolher sempre a opção que desagrade o menor número de pessoas possível.

O método Condorcet permite que os eleitores graduem as opções de acordo com o seu grau de preferência por cada uma delas. Dessa forma, os membros do Debian acreditam que é possível escolher a opção mais aceitável, mesmo que essa não seja a mais votada.

Para que possamos visualizar melhor o processo de votação, vamos simular uma votação para a eleição do líder do projeto. Para isso utilizarei como base os dados da eleição que ocorreu em 2004, que contava com novecentos e onze (911) desenvolvedores. A cédula de votação deverá conter o nome dos candidatos e a opção padrão, identificada como “nenhum dos candidatos listados”, ou “nenhuma da

opções". Nela o eleitor irá colocar qual a sua primeira opção para líder, qual a segunda e assim sucessivamente, dependendo do número de opções que estiverem disponíveis. O eleitor não é obrigado a classificar todas as opções.

Digamos que ao final da apuração do processo eleitoral, o secretário do projeto tenha chegado à seguinte distribuição dos votos:

opção	190 desenvolvedores	130 dos desenvolvedores	95 dos desenvolvedores	85 dos desenvolvedores
1º	João	Maria	Frederico	nenhuma das opções
2º	Maria	Frederico	Maria	
3º	Frederico	João	João	
4º	nenhuma das opções	nenhuma das opções	nenhuma das opções	

O primeiro passo será verificar se todas as opções alcançaram o quorum, isto é, 3Q de votos, ou no nosso caso 45,2741648183597 votos. Caso alguma das opções não tivesse alcançado esse número, seria automaticamente eliminada.

O segundo passo é verificar a maioria requerida, que neste caso é uma maioria simples, cujo valor é obtido pela divisão do número de votos que o candidato recebeu em relação à opção padrão, pelo número de votos que a opção padrão recebeu em relação ao candidato. O valor dessa divisão deverá ser maior ou igual a um (1).

Por exemplo, quatrocentos e quinze (415) pessoas preferem a candidata Maria à opção padrão, enquanto oitenta e cinco (85) desenvolvedores optam por "nenhuma das opções" à Maria; para verificarmos se Maria obteve a maioria exigida, fazemos o seguinte cálculo 415 ÷ 85 = 4,882352941. Como o resultado foi maior que um (1) Maria continua na disputa, caso contrário seria eliminada.

O passo seguinte da apuração será separar e comparar a vitória dos pares opostos. Na tabela seguinte [A] indica os eleitores que preferiram o candidato listado no subtítulo da coluna ao candidato listado no subtítulo da linha e [B] indica os eleitores que preferiram o candidato listado no subtítulo da linha ao candidato listado no subtítulo da coluna :

	A				
		João	Maria	Frederico	Nenhuma das opções
B	João		[A]225 [B]190	[A]225 [B]190	[A]85 [B]415
	Maria	[A]190 [B]225		[A]95 [B]320	[A]85 [B]415
	Frederico	[A]190 [B]225	[A]320 [B]95		[A]85 [B]415
	Nenhuma das opções	[A]415 [B]85	[A]415 [B]85	[A]415 [B]85	

Temos então a seguinte situação:

Candidatos	Vencedor
João (190) contra Maria (225)	Maria
João (190) contra Frederico (225)	Frederico
João (415) contra nenhuma das opções (85)	João
Maria (320) contra Frederico (95)	Maria
Maria (415) nenhuma das opções (85)	Maria
Frederico(415) contra nenhuma das opções (85)	Frederico

Para que possamos chegar ao vencedor são comparados os números de vitórias e derrotas de cada uma das opções, como se segue a tabela abaixo:

Candidato	Vitórias	Derrotas
João	1	2
Maria	3	0
Frederico	2	1
nenhuma das opções	0	3

Mesmo João tendo mais votos, Maria será a nova líder do projeto pois a maior parte dos desenvolvedores de nosso exemplo, a preferem em relação aos outros candidatos. Nesse exemplo, Maria seria vitoriosa em qualquer método de votação Condorcet. No entanto, existem situações em que nos deparamos com o chamado "paradoxo de Condorcet", isto é, quando existe uma ambigüidade, que não pode ser resolvida pela oposição das vitórias e derrotas de cada opção. Digamos que nós tivéssemos apenas três (3) opções, A, B e C e que cada uma delas tenha uma vitória e uma derrota como na tabela a seguir:

Opção	210 desenvolvedores	120 dos desenvolvedores	170 dos desenvolvedores
1º	A	C	B
2º	B	A	C
3º	C	B	A

A oposição dos pares ficará da seguinte forma:

Candidatos	Vencedor
A (330) contra B (170)	A
B (380) contra C (120)	B
C (290) contra A (210)	C

Nesse caso, em que cada uma das opções obteve uma vitória, é eliminada a vitória mais fraca, isto é "C contra A". Logo, A vence B e B não vence ninguém, já que C foi eliminado. Dessa forma, A é a opção vencedora.

A pesar dessa estrutura altamente requintada para a escolha das esferas de decisão do Debian reforçar a imagem idealizada de comunidade, na qual os desenvolvedores estão inseridos em um ambiente de colaboração e de relações horizontais, ela não elimina a disputa inerente ao campo *hacker*. É preciso lembrar que os únicos que têm o direito de votar e de serem votados são os *hackers* que já alcançaram o patamar de desenvolvedores oficiais do Debian e, o processo para se auferir este título a um colaborador projeto, pode levar de dois a três anos, dependendo do número e da qualidade das contribuições feitas pelo candidato.

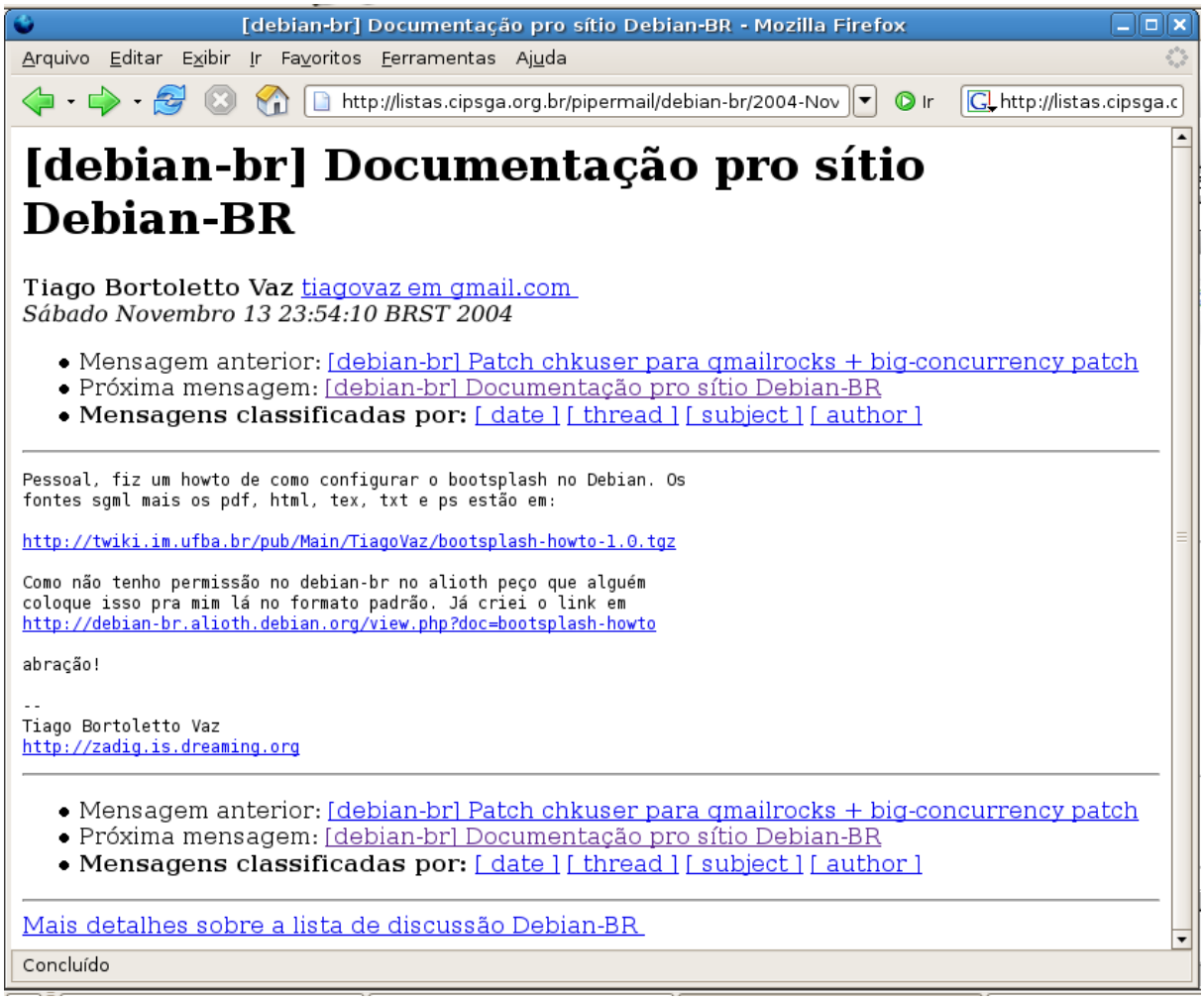
Dessa forma, podemos concluir que a busca do consenso para comunidade Debian, é na verdade, a busca do consenso entre os membros de maior distinção dentro do campo.

O segundo bloco que compõe a comunidade Debian é formado pelos colaboradores. Até o fechamento dessa pesquisa a distribuição Debian era composta por mais de dez mil (10.000) pacotes, distribuídos entre programas, bibliotecas e documentos. Como foi apontado anteriormente, para manter todos estes pacotes atualizados e funcionais, é imprescindivel a ajuda de muito mais pessoas do que apenas os desenvolvedores oficiais do projeto; são necessários esforços dos colaboradores e dos usuários da distribuição.

Uma boa parte dos pacotes que compõem a distribuição são mantidos ou

recebem algum tipo de ajuda de colaboradores, que não são desenvolvedores oficiais do Debian. Estes voluntários podem contribuir tão intensamente com o Debian, quanto um desenvolvedor oficial do projeto, contudo, eles não têm permissão de enviar suas contribuições diretamente aos servidores do Debian. Dessa forma, para que as contribuições dos colaboradores possam entrar nos servidores do Debian, cada colaborador tem um "padrinho", uma espécie de tutor, que se responsabiliza por acompanhar o trabalho do colaborador, revisar os seus códigos e auxilia-lo em possíveis correções. O tutor necessariamente é um desenvolvedor Debian e apenas ele tem permissão para subir arquivos nos servidores do projeto.

Abaixo vemos um exemplo de como um colaborador pode contribuir com o Debian. Na imagem, podemos observar uma mensagem postada na lista de discussões do Debian-BR, um grupo de usuários do Brasil.



Na imagem seguinte, vemos a resposta, postada por Gustavo Noronha, um dos desenvolvedores Debian do Brasil.

O exemplo em questão nos mostra uma das maneiras que os colaboradores podem encontrar para ter suas propostas incluídas nos repositórios do Debian, que é postá-las nas listas de discussão da comunidade ou dos projetos específicos nos quais ele quer intervir.

No caso de nosso exemplo, a colaboração foi aceita e incluída no *site* de documentações do Debian-BR. No entanto, ela também poderia ter sido barrada ou

mesmo receber uma pequena correção antes de ser incluída no *site*. O fato é que esse tipo de colaboração faz com que os *hackers* se tornem conhecidos entre os membros da lista, fazendo com que sejam lembrados pela qualidade, ou pela falta dela em suas colaborações.

Dependendo do número de contribuições de um *hacker*, e da qualidade de suas intervenções nos fóruns, seja postando manuais, pequenas correções para programas ou mesmo tirando dúvidas de usuários menos preparados, esse indivíduo pode dar início ao processo que o levará a se tornar um desenvolvedor oficial do projeto Debian; para isso, deverá ser "apadrinhado" por um indivíduo que já detém este título e que irá acompanhar as suas contribuições e coloca-las no repositório do Debian, quando este for o caso.

O processo para que um colaborador passe à categoria de desenvolvedor oficial, pode levar de dois a quatro anos, dependendo das contribuições que ele faz para o projeto. Quando o tutor acha que seu pupilo está pronto para se tornar um desenvolvedor oficial, encaminha a requisição aos delegados do projeto, que analisam as contribuições desse colaborador e o submetem a uma prova, que abrange principalmente seus conhecimentos técnicos e seu domínio sobre a política do Debian.

Na ponta de todo esta estrutura está o último bloco de nossa análise que são os usuários. Estes últimos não necessariamente têm conhecimentos suficientes para programar, ou mesmo intervir diretamente no código de um programa; porém, indiretamente, acabam influenciando os rumos dos projetos que são encampados pelos dois blocos anteriores, isto é, pelos desenvolvedores e pelos colaboradores.

Freqüentemente, os usuários entram em fóruns para relatar dificuldades em configurar um determinado programa, pedindo ajuda, ou sugestão de *softwares* que executem uma tarefa específica. Também é comum que os usuários encaminhem relatórios reportando erros aos desenvolvedores de um determinado software. Estas pequenas contribuições acabam forçando os desenvolvedores a rever a forma como

estão produzindo seus programas, corrigindo erros e tornando-os mais acessíveis para os usuários comuns.

A relação de simbiose entre os três blocos faz com que a comunidade Debian se organize como um grande organismo vivo, em constante desenvolvimento. Os fatores que permitem que isto aconteça são justamente o trabalho colaborativo e a intensa troca de conhecimento que acontece entre os blocos. Mesmo quando interagem com o o bloco dos usuários, que detêm o menor nível de conhecimento, colaboradores e desenvolvedores estão fazendo com que o conhecimento circule e se desenvolva mais rapidamente. Por exemplo, quando um desenvolvedor posta uma resposta a um usuário em um fórum de discussão, não só teve de pesquisar para solucionar aquele problema, como o usuário acabou aprendendo um pouco mais sobre o *software* do qual se utiliza. No futuro, esse usuário poderá tirar as dúvidas de outro usuário que estiver enfrentando o mesmo problema e, ao mesmo tempo, ele estará mais apto a reportar novos erros à comunidade e de forma muito mais precisa, o que fará com que todo o desenvolvimento do *software* livre caminhe mais rápido.

Contudo, essa interação e a ambiguidade entre comunidade e campo, não é exclusiva da comunidade Debian; ela é inerente a todo o Movimento de *Software* Livre, como podemos ver, ao analisar as motivações de seus usuários e desenvolvedores.

Em sua dissertação de mestrado, apresentada na Universidade Federal do Rio de Janeiro, o pesquisador Maurício Pires Augusto levantou três tipos de variáveis motivacionais entre os desenvolvedores e usuários de *software* livre no Brasil, a saber, motivações tecnológicas, econômicas e psico-sociais.

O quadro a seguir consiste em uma representação do questionário aplicado pelo pesquisador a cento e sete (107) representantes do Movimento do *Software* Livre no Brasil, sendo eles usuários e desenvolvedores e mostra os fatores que os influenciam a utilizar e desenvolver *software* livre.

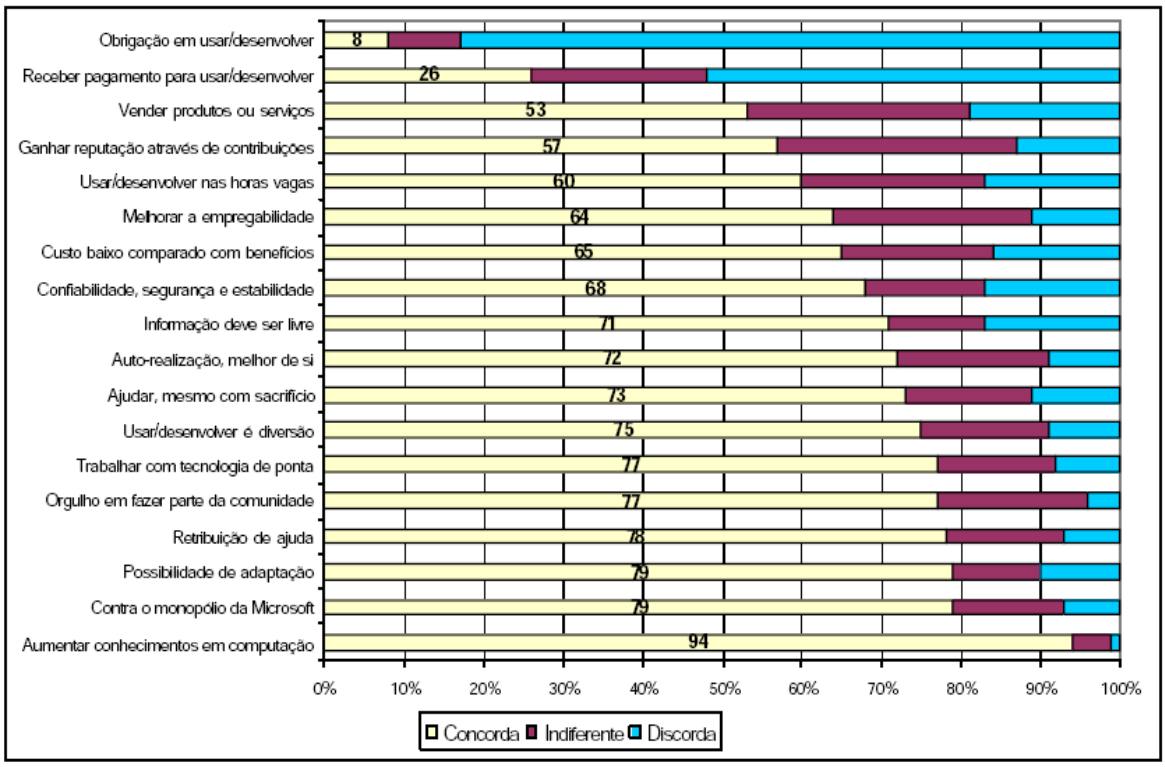


Gráfico 6: Motivações de usuários e desenvolvedores (AUGUSTO - 2003: 73)

Além das questões tecnológicas e econômicas, os fatores psico-sociais levantados por Augusto, refletem alguns aspectos da organização do Movimento *de Software* Livre. A comparação de alguns desses dados mostram como os militantes do *software* livre estão envoltos em uma tensão constante entre uma visão idealizada de comunidade e as relações que se estabelecem dentro de um campo autônomo de produção de conhecimento. Segundo o que foi apurado por Augusto, 77% dos entrevistados sentem orgulho de fazer parte da comunidade, o que está ligado a esta noção idealizada de que a vida em comunidade é sempre uma experiência positiva, de que nela é possível encontrar ajuda e solidariedade de seus pares. Como bem lembra Bauman (2003), essa visão quase sempre se esquece de que a vida comunitária cobra um preço em troca dos benefícios que ela supostamente proporciona.

> *Você quer segurança? Abra mão de sua liberdade, ou pelo menos de boa parte dela. Você quer poder confiar? Não confie em ninguém de fora da comunidade. Você quer entendimento mútuo? Não fale com estranhos, nem*

> *fale línguas estrangeiras. Você quer essa sensação aconchegante de lar? Ponha alarmes em sua porta e câmeras de tevê no acesso. Você quer proteção? Na acolha estranhos e abstenha-se de agir de modo esquisito ou de ter pensamentos bizarros. Você quer aconchego? Não chegue perto da janela, e jamais a abra. O nó da questão é que se você seguir esse conselho e mantiver as janelas fechadas, o ambiente logo ficará abafado e, no limite, opressivo. (BAUMAN - 2002:10)*

Comunidades como o Debian, que têm o seu entendimento construído de forma explícita, por meio de contratos e de resoluções que são votadas pelos seus membros, exigem de seus membros lealdade incondicional, e qualquer desvio é interpretado como uma traição imperdoável. No caso do Movimento de *Software* Livre, a regra de ouro para que este modelo de desenvolvimento se perpetue é que todos devem retribuir à comunidade a ajuda que receberam um dia. Assim, se você quer continuar usando *software* livre e receber auxílio dos membros da comunidade, deve produzir soluções nesses moldes e auxiliar os outros membros sempre que puder. Augusto detecta que esta regra foi incorporada no discurso de 78% dos seus entrevistados, que concordam com a necessidade de retribuir a ajuda que receberam um dia.

Contudo, ao mesmo tempo em que existe o discurso comunitário, as relações entre os membros do Movimento de *Software* Livre se aproximam mais do que Bourdieu chamou de campo, uma vez que o nível de influência de cada membro nos projetos está diretamente ligado ao grau de distinção que eles têm dentro do movimento. Essa busca por distinção é confirmada por 57% dos *hackers* entrevistados, que admitiram que uma de suas motivações para usar e desenvolver *software* livre é o aumento de sua reputação, por meio de suas contribuições à comunidade.

Como foi ressaltado anteriormente, a opção por se detalhar o funcionamento da comunidade Debian foi feita por seu nível organizacional e pelo fato de ela ser mantida exclusivamente com esforços da comunidade, sem se vincular a uma empresa ou

universidade, como ocorre com outras distribuições, apesar de receber doações de equipamentos dessas instituições, que têm interesse que o Debian suporte o seu *hardware* ou simplesmente porque utilizam a distribuição em suas estações de trabalho. Contudo, essa não é a única maneira de se desenvolver *softwares* de código fonte aberto. Josh Berkus, um dos desenvolvedores do *PostgreSQL,* uma das principais soluções livres de banco de dados, aponta pelo menos cinco tipos de desenvolvimento no universo do software livre: solo, monarquia, comunidade, corporativo e fundação. (BERKUS : 2006)

Em janeiro de 2007, o *site* www.sourceforge.org, utilizado pelas comunidades como um repositório de projetos, tinha em seu catálogo 138.299 projetos e 1.474.747 usuários registrados, sendo que todos esses projetos podiam ser classificados dentro de uma dessas cinco classificações ou mediante a combinação de duas delas.

Para chegar a estas classificações, Berkus tomou como base a forma de organização dos projetos, isto é, como são tomadas as decisões críticas, como se dão as contribuições e de que forma acontece o suporte aos usuários.

Ele desenha uma escala que vai do menos ao mais formal, de acordo com a necessidade de organização do projeto. Os projetos solos são, sem dúvida, os menos formais, podendo ser extremamente fácil contribuir com eles, uma vez que seus desenvolvedores se sentem reconhecidos quando alguém se interessa em ajuda-los. Como essas soluções possuem um número reduzido de usuários também é possível para o desenvolvedor dar-lhes mais atenção e incluir sempre que possível as suas solicitações no projeto. As dificuldades que podem ser enfrentadas por um desenvolvedor que queira se juntar a um projeto solo são aquelas referentes à personalidade do mantenedor, que pode ser muito ocupado, dedicando pouco tempo ao desenvolvimento do *software*, ou anti-social, o que é comum se encontrar em *forks*.

Em segundo lugar, no que se refere ao grau de formalidade, estão os projetos monárquicos, com os quais também pode ser extremamente fácil contribuir. Eles são

geralmente originários de um projeto solo que deu certo, e em torno do qual surgiu uma comunidade. As decisões críticas são sempre tomadas pelo líder do projeto ou pelos desenvolvedores que são apontados por ele. Para contribuir com esse tipo de projeto, basta se aproximar do líder, ou de um dos desenvolvedores próximos a ele e enviar seu pedido, ou suas contribuições, que eles decidirão rapidamente se ela será incluída ou não. O tempo para esse tipo de resposta varia de acordo com o tamanho do projeto e quanto seus desenvolvedores estão ocupados. Contudo, como não existem processos de votação, ou longos debates sobre as ações a serem tomadas dentro de um projeto monárquico, o processo é relativamente rápido.

As comunidades, como vimos, são bem mais complexas, já que agregam um número significativo de desenvolvedores que atuam como pares. Algumas votações majoritárias são feitas, mas a maioria das decisões são tomadas a partir da meritocracia e da construção do consenso, o que requer longos debates entre seus membros e qualquer decisão executiva que não respeite as regras da comunidade podem gerar desde longas discussões nas listas de *e-mail* da comunidade até o ostracismo do membro que tomou aquela decisão.

Uma outra forma de se produzir *software* livre é o modo corporativo.

Os projetos corporativos consistem geralmente no código fechado que foi aberto por uma companhia mas não se alienou completamente dele. Para muitos, pode ser duro dizer a diferença entre o projeto e a companhia: a maioria dos programadores são empregados da companhia e o departamento do marketing da companhia determina o sentido estratégico para o projeto. Às vezes estes projetos consistem em código antigo ou não comercializável que a companhia lançou na esfera pública por razões estratégicas ou de relações públicas. Outros projetos incorporados são parte de uma classe crescente das companhias pequenas que vêem o código aberto como o melhor método de distribuição para seus produtos: as companhias com "licenciamento duplo". (BERKUS : 2006)

Contribuir para projetos corporativos é quase sempre muito difícil para os membros do Movimento de *Software* Livre, na maioria das vezes é preciso atravessar uma série de barreiras burocráticas para mandar uma contribuição. Além disso, os desenvolvedores independentes que contribuem não são levados em conta nos processos decisórios e nem podem influir nos rumos do projeto, muitas vezes abrindo mão de seu código em nome da empresa.

Por fim, temos as fundações que são, segundo Berkus, o modelo mais formal de contribuição ao Movimento de *Software* Livre. As fundações desse tipo surgem principalmente de três maneiras: a primeira é quando um projeto de comunidade bem estabelecido sente a necessidade de ter as vantagens de uma estrutura legal por trás de suas ações e de empregar pessoas que possam se dedicar em tempo integral ao projeto. Existem fundações que são criadas em torno de um projeto que é crítico a várias companhias grandes, que utilizam a estrutura formal da fundação, para proteger seus interesses e terem uma voz ativa nos rumos do projeto. Por fim, algumas fundações são o resultado de companhias que se alienam de um projeto corporativo, abrindo seu código para a comunidade e criando uma fundação não comercial para protegê-lo.

É importante ressaltar que os projetos não são estáticos em sua forma de organização. Um projeto que começa como solo pode ir evoluindo aos outros estágios aqui descritos. A própria comunidade Debian já é descrita por muitos como um projeto de comunidade, mas que, devido a seu alto grau de complexidade e organização, começa a apresentar características de uma fundação.

Capítulo – 3

11 Conhecimento : proteção versus aprisionamento

Existem dois grandes marcos na produção do conhecimento colaborativo, especialmente no que se refere à produção de *softwares:* o primeiro é o marco prático e o segundo é o marco legal.

Para que se possa realmente produzir algo de forma colaborativa, é preciso que haja uma troca intensa de idéias e para que isso ocorra na produção de um programa de computador, é necessário que o autor coloque todas as informações do projeto à disposição dos usuários e desenvolvedores que possam estar interessados em utilizar ou aprimorar seu programa. Para isso, são utilizados *sites* que funcionam como repositórios de projetos, como é o caso do www.sourceforge.org, que agrega centena de milhares de projetos de tecnologia aberta; lá é possível encontrar uma breve descrição sobre o *software,* suas funcionalidades, seu estágio de desenvolvimento, além de toda a documentação e os códigos fontes necessários para que se possa auxiliar no desenvolvimento do programa.

No entanto, o marco prático só é possível com o marco legal. Este é determinado pelos tipos de licença que são utilizados pelos desenvolvedores, é o que dirá como podemos proteger o conhecimento sem aprisioná-lo. Para entendermos o quanto o Movimento de *Software* Livre avançou nesse aspecto, é necessário que voltemos um pouco no tempo a fim entendermos qual a origem de conceitos como propriedade intelectual, direito autoral e o que são as patentes. Dessa forma, poderemos deixar claro qual a função desses conceitos, como se desenvolveram no decorrer dos séculos e em que medida cumpriram, ou se desviaram de sua missão original.

Segundo a Organização Mundial de Propriedade Intelectual, esta se estende a

todas as criações da mente, podendo ser dividida em propriedade industrial e direito de autor. Na propriedade industrial, estão incluídas as patentes de invenções, as marcas, os desenhos industriais e as indicações geográficas, enquanto o direito de autor inclui a proteção de obras literárias e artísticas, tais como poemas, romances, obras de teatro, películas cinematográficas e peças musicais. Os programas de computador também são protegidos pelo direito de autor.

Como aponta Imre Simon, *a idéia básica subjacente ao conceito de propriedade intelectual é que o autor ou criador do novo bem determina, dentro de limites socialmente aceitos e legalmente protegidos, as condições sob as quais o bem pode ser usado por terceiros* (SIMON – 2000 : 02). Contudo, os limites do que pode ser socialmente aceito está longe de ser um valor universal e acaba variando de acordo com cada sociedade e com cada época.

Apesar de os primeiros tratados sobre propriedade intelectual serem confeccionados no século XVIII, já era possível ver a preocupação com a defesa dos direitos do autor nesse alvará de 1572 que concede a Luiz de Camões exclusividade sobre a reprodução de *Os Lusíadas*.

Ev el Rey faço faber aos que efte Aluara virem que eu ey por bem & me praz dar licença a Luis de Camoes pera que poffa fazer imprimir nefta cidade de Lisboa, hua obra em Octaua rima chamada Os Lufiadas, que contem dez cantos perfeitos, na qual por ordem poetica em verfos fe declarão os principaes feitos dos Portuguefes nas partes da India depois que fe defcobrio a nauegação pera ellas por mãdado del Rey dom Manoel meu vifauo que fancta gloria aja, & ifto com priuilegio pera que em tempo de dez anos que fe começarão do dia que fe a dita obra acabar de empremir em diãte, fe não poffa imprimir ne vender em meus reinos & fenhorios nem trazer a elles de fora, nem leuar aas ditas partes da India pera fe vender fem liceça do dito Luis de Camoes ou da peffoa que pera iffo feu poder tiuer, fob pena de que o contrario fizer pagar cinquoenta cruzados & perder os volumes que imprimir, ou vender, a metade pera o dito Luis de Camões, & a outra metade pera quem os acufar. [...] E efte meu

Aluara fe imprimirà outrofi no principio da dita obra, o qual ey por bem que valha & tenha força & vigor, como fe foffe carta feita em meu nome por mim afsinada & paffada por minha Chancellaria [...]. Gafpar de Seixas o fiz em Lisboa, a xxiiij : de Setembro, de M.D.LXXI. Iorge da Cofta o fiz efcreuer.

A primeira ocorrência de uma lei de *copyright* é de 1710, quando o parlamento britânico publica o Estatuto de Anne. O estatuto determinava que todas as obras publicadas a partir daquele ano receberiam proteção de *copyright* por quatorze (14) anos, que poderia ser renovada por mais uma vez, caso o autor estivesse vivo. As obras publicadas antes do Estatuto de Anne receberiam proteção de copyright por mais vinte e um (21) anos, sem direito à renovação. De acordo com o parlamento britânico, uma obra deveria se tornar domínio público em no máximo vinte e oito (28) anos. (LESSIG 2006:97)

Em 1710, a noção de *copyright* era muito diferente da que temos hoje; ele se restringia a dizer quem tinha o direito de fazer cópias de um determinado livro. Dessa forma, ao se conceder um direito de *copyright,* o parlamento britânico estava concedendo o monopólio de impressão de uma determinada obra. Nessa época havia grande preocupação com o monopólio de publicações, e o Estatuto de Anne servia para regular o tempo de duração desses direitos. Assim, o Estado deveria proteger o direito exclusivo de publicação, mas só enquanto ele fosse benéfico para a sociedade.

Segundo Lessig, no período em que é redigido o Estatuto de Anne,

> M*uitos acreditavam que o poder que os livreiros exerciam sobre a disseminação do conhecimento estava prejudicando-a, justo na época em que o Iluminismo estava ensinando-os a importância da educação e da divulgação do conhecimento. A idéia de que o conhecimento deveria ser livre era uma marca desse período, e esses interesses comerciais estavam interferindo na idéia*. (LESSIG 2006:80)

Em 1710, quando se concedia o *copyrigth* de uma obra a uma pessoa ou empresa, o Estado concedia o monopólio de impressão dessa obra. Não existia uma

regulamentação de obras derivadas, o que significava que o detentor do monopólio não poderia interferir como sua obra seria interpretada, caso ela fosse adaptada para o teatro, tão pouco receberia qualquer quantia pela adaptação.

Se por um lado o espólio de Sheakespeare, que estava protegido pelo Estatuto de Anne, de 1710, não faturou um centavo pelas milhares de interpretações que a obra do autor teve no teatro, hoje J.K. Rowlings recebe por cada produto derivado de sua obra *Harry Potter*, como peças de teatro, filmes, seriados de TV, bonecos dos personagens, ou revistas em quadrinhos. Qualquer um que queira utilizar a marca *Harry Potter*, deve pagar por isso.

Esse avanço do *coppyright* sobre as obras derivadas se deu graças ao fortalecimento da noção de direito natural, que hoje é amplamente aceita na propriedade intelectual. Como resalta Tachinardi:

> *Propriedade intelectual consiste em direitos associados aos bens e valores imateriais produzidos pela inteligência do homem. A justificativa tradicional para os direitos de propriedade intelectual baseia-se no conceito de justiça. Para a teoria dos direitos naturais – uma das mais antigas fundamentações, que data do século XVII – o homem tem o direito de propriedade natural sobre suas idéias que não podem ser apropriadas por outros. As pessoas têm o direito de receber uma recompensa pelos seus serviços prestados à sociedade. Esta reconhece esses direitos naturais e aceita a obrigação de compensar os inovadores e inventores.* (TACHINARDI – 1993 : 73,74)

A noção de direito natural não era tão aceita em 1710 pelos britânicos devido a uma experiência longa e ruim com "direitos exclusivos" cedidos pela Corte. Como lembra Lessig, os ingleses passaram por uma guerra civil, em parte pelas práticas da Coroa de sustentar monopólios desse tipo muitas vezes sobre obras que já existiam, como por exemplo a impressão da bíblia e a produção de baralhos.

> *Desse modo o copyright, quando visto como um direito de monopólio,*

era claramente visto como um direito que deveria ser limitado. Por mais convincente que o apelo de que "é minha propriedade, e eu devo ter ela para sempre" pareça, ele é tão convincente quanto "é o meu monopólio e deveria ser assim para sempre". (LESSIG – 2006:97)

Os primeiros esforços para se criar um modelo internacional de regulamentação dos direitos de propriedade intelectual ocorrem com as convenções de Paris e de Berna. A convenção de Paris, de 1883, versava sobre a propriedade industrial e a de Berna, de 1886, sobre a propriedade artística e literária. As duas convenções, que ainda estão em vigor, são atualmente administradas pela OMPI – Organização Mundial de Propriedade Intelectual, um órgão das Nações Unidas, criado em 1967. Ambas as convenções foram revistas posteriormente, a fim de se adequarem à nova realidade econômica e tecnológica na qual se encontravam os países membros. Dessa forma, a convenção de Paris foi revista em 1963, em Estocolmo e a de Berna, em 1971, em Paris.

Tanto a convenção de Paris, como a de Berna, não eliminam a necessidade de uma legislação nacional dos países membros. Na verdade, as convenções apontam quais são os pontos relevantes que devem ser levados em conta, quando os países membros forem legislar sobre a questão da propriedade intelectual.

Ao aderirem a estas convenções, os países membros não abriam mão da autonomia para definirem o que é uma patente, qual a extensão dos privilégios concedidos pela patente, quais as áreas sujeitas à patenteabilidade, qual a duração da proteção assegurada pelas patentes, quais as obrigações do patenteado e a quais sanções ele está sujeito, caso se comprove o abuso de poder econômico no exercício do monopólio concedido pela patente e finalmente quais as sanções sobre terceiros que infrinjam os direitos de patente.

Até a criação da OMC (Organização Mundial do Comércio), toda a regulamentação internacional de propriedade intelectual foi gerida pela OMPI. Contudo, a OMPI não satisfazia plenamente os interesses dos países desenvolvidos e

suas indústrias, especialmente dos Estados Unidos da América, que viram seus pedidos para o aumento dos padrões de proteção serem sistematicamente negados, pelo voto do grande número de membros da OMPI, que compunham os chamados países em desenvolvimento. Em segundo lugar, existia a dificuldade de harmonização entre as legislações, já que era opcional aos países membros aderir ou não aos vários tratados regidos pela organização.

Por último, e talvez mais grave, a OMPI não possuía, dentro de sua estrutura, um órgão fiscalizador, que pudesse verificar se os Estados membros estavam cumprindo ou não todas as normas estabelecidas pelos tratados por ela administrados.

Essa fragilidade no sistema de proteção da OMPI gerou uma pressão por parte dos países desenvolvidos, que tinham interesse em um sistema de proteção mais rígido e eficiente. O resultado dessa movimentação das nações desenvolvidas foi a inclusão das questões de propriedade intelectual no âmbito do Acordo Geral sobre Tarifas e Comércio (GATT, sigla em inglês), de 1947, durante a Rodada do Uruguai.

Em 1994, a Rodada do Uruguai resulta na criação da OMC e em diversos acordos anexos, entre eles o Acordo Sobre Aspectos dos Direitos de Propriedade Intelectual Relacionados ao Comércio, ou TRIPS (sigla em inglês). Desde então, os direitos de propriedade intelectual vêm sendo regulamentados pelo TRIPS, o que não quer dizer que a OMPI tenha perdido sua função. Na realidade, o TRIPS trabalha em cooperação com a OMPI, incorporando e, a todo tempo, fazendo referências às convenções por ela administradas. Além das convenções de Paris e Berna, são citadas no TRIPS a Convenção de Roma, para a proteção dos artistas intérpretes, produtores de fonograma e organizações de radiodifusão, de 1961 e o Tratado Sobre Propriedade Intelectual em Matéria de Circuitos Integrados (PICI), de 1989, assinado em Washington. A idéia por trás dos TRIPS foi a de estabelecer um padrão mínimo proteção que deveria ser seguido por todos os membros da OMC e dar o primeiro passo em direção a um sistema mundial de patentes.

Segundo Cícero Gontijo, os TRIPS foram sem dúvida o ponto mais difícil da

Rodada do Uruguai, uma vez que:

> *Todos os outros dezesseis acordos propunham a abertura do mercado, redução de tarifas, redução de barreiras alfandegárias. Eram todos no sentido de abertura, de redução de garantias e direitos, visando um mundo que funcionasse como um comércio único. Seguiam todos na direção de derrubar barreiras alfandegárias. O TRIPS ia na contramão: criava barreiras, e, num sentido muito específico, criava barreiras para proteger ainda mais quem já tinha tecnologia, já tinha patente, já estava pesquisando. Estes iriam ganhar o maior poder possível, pois o acordo transformava esses fatores em produtos ainda mais valiosos.* (INSTITUTO DE ESTUDOS SOCIOECONÔMICOS, 2003:26)

Para notarmos como se dão as mudanças do conceito de *copyright* do Estatuto de Anne até chegarmos ao TRIPS, é interessante observarmos como ocorreram as mudanças de prazo e escopo dentro da legislação norte americana sobre os direitos autorais. Estas mudanças na legislação dos EUA também irão influenciar fortemente a forma como as comunidades de *software* livre passarão a licenciar seus produtos.

A primeira lei federal norte-americana sobre *copyrigth,* é de 1790 e é muito similar ao Estatuto de Anne. O prazo de proteção estabelecido é de quatorze (14) anos, sendo possível renova-lo por mais quatorze (14) anos caso o autor estivesse vivo e se ainda fosse de seu interesse manter a proteção do *copyright*.

> *Embora houvessem muitas obras criadas nos Estados Unidos nos primeiros dez anos da República, apenas 5% delas foram realmente registradas no regime federal do copyright. De todas as obras que foram criadas antes de 1790 e entre 1790 e 1800, 95% passaram imediatamente para o domínio público; o equilíbrio poderia chegar ao domínio público em 28 anos no máximo,e normalmente chegava em catorze anos.* (LESSIG : 2006 – 119, 120)

Em 1831, o período de proteção passou de quatorze para vinte e oito anos, somados aos quatorze (14) anos de renovação, uma obra passaria ao domínio público em no máximo quarenta e dois (42) anos. Em 1909, o período de renovação do *copyrigth* também foi ampliado e passou a ser de vinte e oito (28) anos, o que elevou o período máximo de proteção para cinqüenta e seis (56) anos. Nos últimos quarenta e cinco (45) anos, o congresso americano aumentou os prazos de proteção por onze (11) vezes. Até 1976, as mudanças são pequenas, com aumentos variando entre um ou dois anos. No entanto, em 1976, ocorrem duas grandes mudanças: a primeira é o aumento do período dos direitos de copyrigth em dezenove (19) anos e a segunda e mais grave é a retirada da necessidade de renovação do *copyright*. Para as obras criadas a partir de 1978, passou a vigorar um único período de copyright, o período máximo. Para autores "naturais" este período seria o de sua vida mais cinqüenta (50) anos e para corporações seria de setenta (70) anos.

Em 1992, o congresso americano aboliu a necessidade de renovação para todas as obras criadas antes de 1978, e todas que ainda estavam sobre proteção do *copyright* passaram a receber o período máximo disponível. Em 1998, com a Lei de Extensão do Período de Copyright Sonny Bono (*Sonny Bono Copyright Term Extension Act*), aumentou-se, em mais vinte (20) anos a proteção dos *copyrights* existentes e futuros. Dessa forma, o período de proteção passou a ser de noventa e cinco(95) anos.

Depois de todas essas mudanças de prazo, apesar de a lei americana estabelecer que o monopólio concedido pelo *copyright* deve ser garantido por um tempo limitado, já não está claro se algum dia será possível, de fato, passar novas obras para o domínio publico, já que a todo momento os prazos de proteção são ampliados.

Além das mudanças no prazos, o *copyright* também passou por várias mudanças de escopo. Tanto o Estatuto de Anne, de 1710, como a primeira legislação norte americana, de 1790, legislavam sobre aspectos muito restritos, ou seja, a publicação de livros, mapas e gráficos. Estavam fora da proteção por exemplo as obras musicais e arquitetonicas.

O *copyright*, em 1710, concedia o monopólio para a publicação de um determinado livro. Era uma lei restrita, que regulava as ações de uma parcela restrita da sociedade, a saber, os autores, os livreiros e as editoras.

Porém, isso mudou de forma radical e, atualmente, o *copyrigth* protege igualmente livros, musicas, obras cinematográficas e arquitetônicas. Contudo, o aspecto que mais nos interessa nessa mudança é que, agora, além de ampliar o seu espectro de proteção, o *copyrigth* garante direitos sobre as obras derivadas que forem "significativamente" baseadas em uma obra protegida.

Esse mecanismo foi desenvolvido a princípio para evitar que alguém publicasse um livro e meses depois outro indivíduo publicasse o mesmo livro com alterações insignificantes e o registrasse como uma obra completamente nova. Contudo, essa alteração concedeu poderes muito maiores ao detentor do *copyright*, já que lhe dá, não só o monopólio de publicação, mas também o direito sobre peças de teatro, obras cinematográficas, traduções, resumos ou qualquer tipo de produtos que possam vir a ser desenvolvidos a partir de sua obra, escapando a isso apenas uma pequena margem do chamado " uso justo".

O uso justo, porém, é bem controverso e na prática se aplica às citações em trabalhos acadêmicos, em resenhas ou críticas que são escritas sobre as obras.

Apesar de os países desenvolvidos terem sido vitoriosos ao aprovarem os TRIPS em sua busca por um sistema mundial de patentes, um sistema desse tipo ainda encontra muita resistência, principalmente por parte dos países em desenvolvimento que seriam os principais prejudicados com uma medida dessas.

Durante os anos 80 do século XX, a literatura sobre a importância de um sistema de proteção de patentes desaconselhava fortemente o reconhecimento de patentes internacionais por parte dos países em desenvolvimento. Segundo Constantine Vaitsos:

> *Os sitemas de patentes em países em desenvolvimento tem um efeito negativo predominante e resulta em poucos benefícios para esses países: virtualmente, as patentes quase sempre controladas por grandes corporações*

estrangeiras, são usadas como veículo para se alcançar os privilégios dos monopólios, o que não contribui para os investimentos estrangeiros e para o fluxo de tecnologia em direção às nações em desenvolvimento porque limita o seu avanço tecnológico à imitação e à adaptação. (VAITSOS– 1973 : 71 apud TACHINARDI – 1993 :81)

Contudo, os TRIPS eram apenas o primeiro passo; à medida em que os EUA foram encontrando dificuldades em consolidar um sistema único de proteção de patentes, passaram a atuar em múltiplas frentes a fim de complementar através de acordos bilaterais, os pontos que estavam em aberto nos TRIPS.

Um ótimo exemplo da busca por complementariedade dos EUA pode ser visto no que diz respeito ao artigo 27.3 item B dos TRIPS, no qual se lê:

3 - Os Membros também podem considerar como não patenteáveis:

b) plantas e animais, exceto microorganismos e processos essencialmente biológicos para a produção de plantas ou animais, excetuando-se os processos não biológicos e microbiológicos. Não obstante, os Membros concederão proteção a variedades vegetais, seja por meio de patentes, seja por meio de um sistema "sui generis" eficaz, seja por uma combinação de ambos. O disposto neste subparágrafo será revisto quatro anos após a entrada em vigor do Acordo Constitutivo da OMC.

A decisão de rever este artigo quatro anos depois, evidencia a insatisfação tanto dos países em desenvolvimento, quanto das nações desenvolvidas. Os primeiros não concordavam com o subparágrafo em questão basicamente por dois motivos: ou por saberem que seus objetivos de crescimento econômico e industrial seriam afetados negativamente, com a aprovação desses padrões, ou por estarem em desacordo com o patenteamento de seres vivos. Os últimos por desejarem eliminar, com este e outros artigos, todo tipo de exceção podendo ampliar ainda mais os direitos de propriedade intelectual.

Os EUA passaram, então, a firmar Tratados de Livre Comércio (TLC) de forma

bilateral ou com blocos econômicos, que forçam os signatários a assinarem outros tantos tratados internacionais de propriedade intelectual regidos pela OMPI. Como é o caso do TLC EUCADR, o Tratado de Livre Comércio entre os Estados Unidos, América Central e República Dominicana, que entre outras coisas exige dos signatários que ingressem na UPOV-91[6], para a proteção de variedades de plantas, além da ratificação de outros sete acordos internacionais de propriedade intelectual, e do esforço para a assinatura e outros três, caso ainda não o tenham feito. (CERVANTES – 2005)

O argumento central em torno do aprisionamento cada vez maior do conhecimento está baseado na afirmação de que, sem direitos de propriedade intelectual não existe inovação. Mesmo que não haja dados empíricos que comprovem esta afirmação, os detentores de patente defendem que caso seu trabalho não fosse protegido, os inventores, pesquisadores e artistas não teriam como manter um nível de vida digno e não receberiam o estímulo necessário para continuar inovando.

De fato temos vários exemplos que contradizem esta afirmação, sendo que um deles é o próprio Movimento de S*oftware* Livre. Assim como é simplista pensar que a única motivação possível para a inovação sejam os benefícios econômicos concedidos ao pesquisador, é igualmente limitado pensar que apenas a boa vontade e o espírito de cooperação dos desenvolvedores, poderiam gerar a consolidação do modelo colaborativo de produção do conhecimento.

Como vimos no capítulo anterior, existem várias outras motivações que agem sobre os *hackers* que compõem este movimento; alguns o fazem para melhorar seu currículo, outros para aprimorar seus conhecimentos em informática; há os que o fazem por motivos econômicos e outros simplesmente para construírem algo do que possam se orgulhar. No entanto, nenhuma dessas motivações é excludente e o mais

6 Criada, em 1961, por países europeus o Convênio UPOV, em sua ata vigente, de 1991, provê um marco de lei de propriedade intelectual às variedades de plantas, muito semelhante às patentes. Além disso, a UPOV-91 permite a dupla proteção, pois uma mesma pessoa ou empresa pode pedir a proteção junto à UPOV e junto à legislação de patentes.

comum é que atuem de forma complementar na produção do *software* livre. Além disso, não se pode esquecer que o Movimento de *Software* Livre propõem não só uma nova forma de produzir o conhecimento, mas também uma forma de protege-lo e obter lucro com ele.

Como vimos anteriormente, em vez de receber pelas cópias vendidas de seus programas, os *hackers* recebem por serviços prestados; implantando e adaptando suas soluções, ou de terceiros, às necessidades dos seus clientes. A ambivalência entre o campo *hacker* e a noção de comunidade que eles defendem, faz com que o campo se torne uma arena na qual é disputado o título de maior colaborador da comunidade. Essa disputa é o que retroalimenta o movimento, já que, ao colaborarem, os *hackers* aumentam o seu grau de distinção dentro do campo e quanto maior o grau de distinção desse profissional, mais caro é o valor das suas horas de trabalho. Assim, a lógica de trabalho desses intelectuais nos dirá: quanto maior for o número de pessoas colaborando, menor será o custo de desenvolvimento e maior será a sua velocidade, por isso não se deve aprisionar o conhecimento; quanto maior o número de colaborações, maior o grau de distinção obtido pelo *hacker* dentro do campo; quanto melhor a reputação, maior o valor cobrado pelas horas trabalho e, provavelmente, maior o número de oportunidades profissionais recebidas por estes *hackers;* por isso, é necessário contribuir o máximo possível com a comunidade e sempre que possível manter-se em evidência no campo.

Os defensores das patentes poderiam dizer ainda que o caso dos *hackers* é isolado e que isso não funcionaria se o exemplo fosse transposto para o mundo corporativo. Poderiam dizer, ainda, que é impossível o avanço da pesquisa em outros setores sem que haja a proteção necessária, o que tornaria os empreendimentos economicamente inviáveis. No entanto, mesmo que nós ignorássemos por completo as ações de grandes empresas de tecnologia, como a *IBM*, *Novel*, e *Sun* entre outras, que vêm investindo cada vez mais em projetos de tecnologia aberta, para contar com o auxílio das comunidades de *software* livre e assim, diminuir os custos de pesquisa e

desenvolvimento, Tachinardi nos traz o exemplo das indústrias químicas e farmacêuticas no Brasil, no período de 1970 à 1990. Segundo a autora:

> *Entre os anos de 1970 e 1990, os investimentos estrangeiros diretos na área química e farmacêutica cresceram, respectivamente, quase oito e mais de treze vezes, num ritmo, em ambos os casos, muito mais acelerado do que o investimento estrangeiro em outras áreas em que já era possível pleitear proteção patenteária. São números que indicam que os investidores estrangeiros não se sentiram desestimulados a aplicar no Brasil no setor e que, ao trazer tecnologia para produzir no país, encontraram formas satisfatórias de remuneração tanto para suas aplicações de capital em capacidade produtora no Brasil quanto para gastos incorridos nos países de origem em pesquisa e desenvolvimento, ainda que não seja permitido às filiais pagar às matrizes royalties por patentes.* (TACHINARDI – 1993 :73)

Mesmos sem dados empíricos capazes de comprovar a eficácia do aumento da proteção de direitos de propriedade intelectual, sobre a produtividade de invenções e inovações, parece que os governos caminham de olhos fechados para a total privatização do conhecimento. A cada nova ampliação do escopo de proteção dos direitos de patente, avançamos mais na direção da privatização do conhecimento e restringimos cada vez mais as possibilidades de pesquisas futuras.

James Boyle, que não é contra os direitos de propriedade intelectual, mas defende um mecanismo mais equilibrado e que garanta a defesa do domínio público, chama-nos a atenção para o fato de que a cada ano a abrangência das patentes se estende para cobrir as "idéias", o que há vinte anos atrás, os acadêmicos pensavam ser não patenteáveis. O autor aponta, como os exemplos mais óbvios, o patenteamento de métodos de negócio, que abarcam a "invenção" de métodos de venda ou de contabilidade.

No mundo do *software*, esse patenteamento de idéias pode significar a proteção

de funcionalidades. Por exemplo, em vez de se proteger o trecho do código fonte, que acrescenta negrito às palavras, eu poderia proteger a função negrito, e mesmo que um *hacker* escrevesse outro código para realizar a mesma tarefa, ainda sim ele estaria infringindo as leis de patente. É como se em vez de se proteger a letra de uma determinada música de amor, concedessem a patente das músicas de amor a um único compositor, logo, todos que quisessem escrever musicas desse gênero seriam obrigados a pagar *royalties* a ele.

O possível patenteamento das "idéias" é uma das coisas que mais ameaçam o Movimento do S*oftware* Livre. Em 2006, a União Européia vetou uma proposta de patenteamento de *softwares,* que atualmente são protegidos pelo direito de autor, tal qual as obras literárias. No entanto, essa discussão ainda não se encerrou completamente; uma mudança desse tipo na legislação, poderia colocar algoritmos e certas características fora do alcance do *software* livre por até vinte anos.

Segundo Richard Stallman:

> *existem algumas maneiras de se tratar com as patentes: podemos buscar evidência de que a patente não é válida, e podemos buscar maneiras alternativas de realizar o trabalho. Porém, cada um desses métodos funciona apenas certas vezes. Quando ambas falham, uma patente pode forçar que todo software livre careça de alguma característica que os usuários desejam* (STALLMAN – 2005 :174)

Devemos atentar para o fato de que um algoritmo não é um *software;* na verdade, ele é uma seqüência lógica de tarefas que podem ser executadas por um computador, ou mesmo por um ser humano. Patentear um algoritmo é muito mais grave que patentear um *software,* é como patentear uma idéia que pode vir a ser um *software.* Na matemática, o algoritmo é constituído pela seqüência de processos, e símbolos que os representam, utilizados para efetuar um cálculo. Buscar proteção patenteária para tal abstração é como tentar privatizar a própria matemática.

Este tipo de patente traria possibilidades de concentração de conhecimento e

poder sem precedentes. Imaginemos o caso de um estudante que tenha desenvolvido um editor de textos que traz como inovação a possibilidade de acrescentar negrito e itálico às palavras. Suponhamos ainda, que a função negrito já estivesse patenteada por alguma empresa. Nesse caso, a empresa poderia impedir que o estudante colocasse o seu produto no mercado, a menos que ele pagasse os *royalties* da funcionalidade em questão ou, caso fosse de seu interesse, poderia comprar os direitos sobre a funcionalidade itálico. Caso a empresa comprasse a funcionalidade itálico, quando surgisse um novo editor de textos capaz de acrescentar às palavras negrito, itálico e sublinhado, os donos do novo *software* serião obrigados a lhe pagar pelas duas funcionalidades ou lhe vender a terceira. Com o tempo, esse tipo de proteção tornaria impossível desenvolver qualquer tipo de *software* sem que fosse necessário pagar altas taxas com *royalties* a um grupo cada vez menor de empresas, encarecendo a pesquisa, atrasando o desenvolvimento e concentrando o conhecimento.

Por mais absurda que possa parecer, esse tipo de proteção patenteária já vem sendo concedida nos EUA há alguns anos, e foi o que permitiu à Microsoft patentear o algoritmo que ativa uma aplicação com o duplo *click* do *mouse*. Dessa forma, toda vez que abrimos um programa em nossos computadores e utilizamos o duplo *click* para isso, estamos fazendo uso de uma patente da gigante dos *softwares*.

Os pedidos de ampliação de prazos e escopo das patentes e do *copyright* nunca andam só, eles são sempre acompanhados das exigências de mecanismos mais eficazes de controle. Como lembra Imre Simon, a política de propriedade intelectual, que já é praticada há quase três séculos, encontra duas grandes dificuldades: uma relacionada à tecnologia e outra relacionada aos benefícios que o acesso e a possibilidade de cópia podem gerar à sociedade.

> *Por um lado, a tecnologia de fazer cópias evoluiu constante e substancialmente com o tempo, dificultando a imposição da lei e podendo chegar a torná-la inefetiva. Por outro lado, a cópia tem inúmeros papéis positivos e altamente desejáveis para o progresso das sociedades em geral e*

para a preservação e incremento das suas culturas. Para manter um equilíbrio entre os incentivos à produção intelectual, a pressão da facilidade de fazer cópias e o interesse da sociedade de ser bem suprida de bens de informação essenciais, a lei é atualizada de tempos em tempos de acordo com a situação vigente. (SIMON - 2000:3)

Sem dúvida nenhuma, nas tecnologias da informação, um dos papéis positivos da cópia podem ser encontrados, não só na forma como se organizam as comunidades de *Software* livre, mas também, no caráter hipertextual da própria *web.* O *hipertexto* incita e propõe a veiculação de idéias, por meio da publicação de páginas e *sites* que compõem a Internet. Ao permitir copiar e *linkar* o seu texto com o de outros autores, o *hipertexto* consolida-se como a tecnologia da colaboração e do não isolamento, contrapondo-se à linearidade e a rigidez hierárquica das informações. Contudo, nem todos vêm este tipo de tecnologia como algo positivo; os defensores de um sistema mais rígido de propriedade intelectual vêem, na hipertextualidade e na possibilidade da livre circulação de idéias e do fluxo de arquivos digitalizados pela web, uma ameaça constante a seus interesses.

Segundo o sociólogo Sérgio Amadeu da Silveira:

A web é o repositório da hipertextualidade. Não usar plenamente os recursos do hipertexto e criar mecanismos de proteção de sites contra o acesso livre é, sem dúvida, uma possibilidade. Mas também é a negação das imensas potencialidades que a tecnologia hipertextual assegura e inspira. A recusa dessa inspiração dá-se pela defesa de um comportamento econômico e político consolidado na idéia conservadora de um modelo de propriedade rígido, típico do mundo material e transposto para o universo dos símbolos. Seu objetivo é concentrar riquesas em monopólios de algoritmos e em oligopólios de produção simbólica. Como trata-se de uma conduta que interessa a poucos, para se manter, precisa de um apelo ideológico, uma plataforma ideológica e uma rede econômica e política de sustentação, bem

como necessita "conformar o apoio da opinião pública." (SILVEIRA -2005 : 77)

Na busca por mais proteção, os oligopólios da informação iniciaram uma grande guerra contra a construção coletiva do conhecimento e o livre fluxo das idéias, lançando seus ataques massivos contra qualquer tipo de tecnologia que possa vir a colocar em risco seu velho modelo de negócios e contra a privacidade dos cidadãos. Para isso, campanhas milionárias são desenvolvidas para tentar desestimular o uso dessas tecnologias e justificar os abusos e exageros cometidos por aqueles que pretendem ser os novos donos do conhecimento.

Atualmente a principal campanha dos detentores de patente é a incessante luta contra a "pirataria". Por meio dela tentam engajar governos, empresas e até o cidadão comum na defesa de seus interesses. O argumento central por trás de tal campanha é que a propriedade intelectual é uma propriedade como outra qualquer, e deve ser tratada com tal. Assim como é errado tomar o carro de uma outra pessoa, fazer uma cópia não autorizada de uma música, ou de um programa de computador, também é um roubo e deve ser combatido e denunciado por toda a sociedade.

Este argumento tem um forte apelo ideológico, afinal de contas, que pessoa de boa índole poderia concordar com um roubo?

Com base no argumento de que as cópias não autorizadas são um roubo e na idéia de que quanto mais fácil é o processo de cópia mais rígidos devem ser os mecanismos de controle, os oligopólios da informação abrem verdadeiras guerras jurídicas contra todas as novas tecnologias que coloquem em risco seu antigo modelo de negócios e tentam justificar seus abusos e exageros cometidos contra os cidadãos e sua privacidade.

Um dos exemplos mais emblemáticos dessa guerra dos antigos modelos de negócio contra as novas tecnologias é o caso Napster, um dos primeiros programas que permitiam a troca de arquivos entre internautas, por meio de uma rede P2P, ou *Peer-to-Peer*. O Napster criava uma rede virtual que, apesar de possuir servidores centrais para

controlar ações críticas da rede, diferenciava-se da arquitetura cliente/servidor, na qual alguns computadores são dedicados a servirem dados a outros. A rede P2P estabelecida pelo *software* permitia que todos os usuários servissem e recebessem dados, isto é, da mesma forma como eu poderia me conectar diretamente a um computador na Croácia, para baixar um arquivo de música, ou um manual que estivesse sendo compartilhado pelo internauta croata, várias pessoas de todo o mundo poderiam se conectar ao meu computador, para buscar os arquivos que eu estivesse compartilhando.

Aliado à tecnologia de compactação *Mpeg Layer III,* popularmente conhecida pelos arquivos MP3, que compactam arquivos de áudio em um tamanho razoável para que viajem na Internet, sem que haja grandes perdas na qualidade do som, o *Napster* rapidamente se tornou a mais popular ferramenta de troca de arquivos pela rede, principalmente arquivos de música.

Preocupados com seus direitos de propriedade intelectual, os integrantes da banda norte americana Metallica foram os primeiros artistas a abrirem um processo contra a empresa Napster que fabricava o programa homônimo. Em 13 de abril de 2000, quando o Metallica abriu o processo, seus advogados afirmaram que a banda estava tendo um prejuízo de mais de US$ 10 milhões. O valor era baseado na quantia pedida pela Associação das Gravadoras dos EUA (RIAA) no processo que já vinha movendo contra o Napster, o equivalente a US$ 100 mil por cada música pirateada. Depois do Metallica, a comunidade de músicos em todo mundo se dividiu: alguns contra outros a favor. (VERSIGNASSI- 2000)

Se em 1710, a proposta do *copyright* era regulamentar o uso de uma determinada tecnologia, naquele caso as prensas, concedendo o monopólio da publicação e reprodução de um determinado livro, depois do enrijecimento das leis de propriedade intelectual, houve uma mudança de foco e os detentores de *copyright* passaram a atacar as tecnologias em si e não a maneira como são utilizadas.

Em 2000, o Napster foi retirado do ar, e a empresa proprietária do software foi a

julgamento. Mesmo informando que já havia desenvolvido uma tecnologia capaz de bloquear a transferência de 99,4% do material identificado como ilegal, a corte disse ao advogado do Napster que 99,4% não era suficiente. O Napster deveria eliminar totalmente as violações de *copyright*.

Como aponta o jurista Laurensse Lessig:

> *Se 99,4% não é o suficiente, então essa é uma guerra contra as tecnologias de compartilhamento de arquivos, não uma guerra contra violações de copyright. Não há como garantir que um sistema de P2P vá ser usado o tempo todo dentro da lei, da mesma forma como é impossível garantir que 100% dos videocassetes ou 100% das máquinas Xerox ou 100% das armas de fogo serão usadas dentro da lei. Tolerância zero quer dizer que não teremos P2P. A decisão da corte define que nós como uma sociedade devemos perder os benefícios do P2P, mesmo para os usos totalmente legais e benéficos que ele pode representar, se isso for necessário para garantir que não haverão violações de copyright causadas pelo P2P.* (LESSIG : 2006 – 67)

Contudo, assim que o Napster saiu do ar, vários outros programas foram criados para substituí-lo. Como a segunda geração de programas de compartilhamento não necessitava de servidores centrais para administrar nenhuma de suas transações, tornou-se muito mais complicado atacar as empresas e indivíduos que produziam estas ferramentas. Dessa forma, a indústria do entretenimento, principal prejudicada por esse tipo de tecnologia, passou a processar cidadãos comuns, para intimidar os usuários desses *softwares* por meio do efeito demonstração.

Nos anos que se seguiram após o fechamento do Napster, notícias como a da inglesa Silvia Price, passaram a ser comuns nos cadernos de tecnologia dos jornais e principalmente nos noticiários on-line. Silvia, que não sabia operar o computador, havia recebido uma multa de quatro mil libras, porque sua filha de 14 anos, tinha mais de 1,4 mil músicas em um computador, que ficava ligado vinte e quatro horas a redes

de compartilhamento P2P. A filha de Silvia, por sua vez, dizia não imaginar que estivesse cometendo um crime, já que todos seus amigos do colégio faziam o mesmo. Na época em que a reportagem foi veiculada, 23 de junho de 2005, Silvia alegava não ter o dinheiro necessário para pagar a multa e que isso, provavelmente, resultaria na sua prisão. (TERRA TECNOLOGIA – 2005)

Em 2007, a Rússia, que vem desenvolvendo grandes esforços contra a pirataria para poder integrar-se à OMC, também assistiu a um caso parecido, no qual, o professor Alexander Posonov foi acusado de violar propriedade intelectual da Microsoft, ao usar nos computadores de sua escola, cópias não-licenciadas de programas da gigante do software. O ex-líder soviético, Mikhail Gorbachev, chegou a interceder junto ao fundador da Microsoft, Bill Gates, pedindo clemência pelo professor em uma carta divulgada no site de sua instituição de caridade. Na carta, Gorbachev dizia,

> *Um professor, que dedicou sua vida à educação de crianças e que recebe um salário modesto que nem se compara aos salários de funcionários comuns de sua companhia, está sendo ameaçado de prisão em campos de detenção siberianos [...]. Temos muito respeito pelo trabalho dos programadores da Microsoft e não estamos de maneira nenhuma colocando em dúvida o princípio de punição a quem violar propriedade intelectual [...]. Entretanto, neste caso nós pedimos para você mostrar clemência e retirar a queixa contra Alexander Ponosov [...] Este gesto nobre será recebido de maneira entusiasmada por todos aqueles que usam produtos Microsoft na Rússia.* (REUTERS : 2007)

Quando lemos notícias como estas, é inevitável nos questionarmos como se descobriu que a garota de quatorze anos tinha 1,4 mil músicas em seu computador, ou como a Microsoft descobriu que as cópias utilizadas pelo professor Ponosov eram ilegais. É claro que existe a possibilidade de ambos terem sido denunciados por cidadãos interessados em combater o roubo da propriedade intelectual, por um

amiguinho do colégio ou mesmo por um aluno insatisfeito com as notas ao final do semestre. No entanto, existe ainda a possibilidade da espionagem.

Muito mais sofisticada do que a espionagem da época da guerra fria, a espionagem da sociedade da informação se dá em grande parte, de forma consentida, por meio de contratos aceitos com um *click* do *mouse,* nas licenças de uso dos *softwares* instalados em nossos computadores. Um exemplo desse tipo de espionagem, amplamente aceito e incentivado pela indústria do entretenimento e do *software* proprietário, está transcrito no Acordo de Licença do Usuário Final do *Windows* XP, ou EULA (sigla em inglês), no qual pode se ler no quinto subitem do parágrafo 7:

> *Os provedores de conteúdo estão usando a tecnologia digital de gerenciamento de direitos ("Microsoft DRM") presente no Produto a fim de proteger a integridade de seu conteúdo ("Secure Content") para que sua propriedade intelectual, e direitos autorais não sejam desapropriados. Os proprietários desses Conteúdos de Segurança ("Secure Content Owners") devem, de vez em quando, solicitar que a Microsoft forneça as atualizações referentes à segurança para os componentes do Microsoft DRM do Produto ("Security Updates"), que pode afetar a sua capacidade de copiar, exibir e/ou reproduzir o Secure Content através de um software da Microsoft ou aplicações de terceiros que usam o Microsoft DRM.* ***Você, portanto, aceita que, caso eleja fazer o download de uma licença pela Internet, permitindo que você utilize o Secure Content, a Microsoft pode, em conjunto com tal licença, também fazer o download em seu computador, tal como atualizações de segurança, de que um Proprietário do Secure Content solicitou a distribuição da Microsoft.*** *A Microsoft não irá recuperar quaisquer informações pessoalmente identificáveis ou outras informações, pelo seu computador ao baixar algo como o Security Updates*. (grifo meu)

Em outros parágrafos do EULA a Microsoft da indícios de que existem várias

portas em seu sistema operacional, que o usuário comum não consegue bloquear, através das quais a empresa pode verificar que tipo de conteúdo existe na sua máquina, como vemos nos sub-itens "Consentimentos no uso de Dados" e "Recursos de jogos pela Internet/Atualização", transcritos abaixo:

> *Consentimentos no uso de Dados:* ***Você está de acordo que a Microsoft e suas afiliadas podem coletar e usar informações técnicas reunidas de qualquer forma como parte dos serviços de suporte ao produto oferecido a você****, se houver, relacionados ao Produto.* ***A Microsoft pode usar essas informações somente para aprimorar nossos produtos ou fornecer serviços ou tecnologias customizadas a você. A Microsoft pode abrir essas informações a outros, mas não de maneira que o identifique.*** (grifo meu)
>
> *Recursos de jogos pela Internet/Atualização: Se você optar por usar os recursos de jogos pela Internet ou atualização dentro do Produto, é necessário utilizar determinado sistema de computador, informações de hardware e software para implementar os recursos.* ***Usando esses recursos, você autoriza explicitamente a Microsoft, ou seu agente, para acessar e utilizar informações necessárias para fins de jogos e atualização pela Internet. A Microsoft pode usar essas informações somente para aprimorar nossos produtos ou fornecer serviços ou tecnologias customizadas a você. A Microsoft pode abrir essas informações a outros, mas não de maneira que o identifique****.* (grifo meu)

Seja com a justificativa de proteger a propriedade intelectual ou a de oferecer serviços de melhor qualidade a seus clientes, a Microsoft reserva-se o direito de espionar todos os seus usuários, independente de eles estarem ou não cometendo um ato ilícito. Assim, como todos devem ficar sem os benefícios das redes P2P, para garantir o velho modelo de negócios da indústria fonográfica, todos são obrigados a abdicar de sua privacidade, para defender os interesses dos oligopólios da informação.

Se a Microsoft pode auditar os conteúdos de todos os computadores que têm seu sistema operacional instalado, o que em 2006 significava aproximadamente 97% dos computadores pessoais, quem auditará a Microsoft para saber que fim ela dá a esses dados?

De que serviriam, então, os tais dispositivos de segurança DRM se eles não podem identificar o infrator do *copyright?*

Todos esses abusos são feitos com base na afirmação de que a propriedade intelectual é uma propriedade como outra qualquer e deve ser protegida como tal. No entanto, o que se esquece é de que, assim como afirma a OMPI, as leis de propriedade intelectual dizem respeito aos produtos da mente, istó é, bens intangíveis, que têm características e regulamentações muito distintas dos bens tangíveis.

As leis de propriedade de bens tangíveis, como a posse da terra, por exemplo, são regras que regem a distribuição de bens materiais, escassos, passíveis de desgaste e de uso excludente. A terra é um bem tangível, é possível tocar a terra e até mesmo cercá-la, ela é escassa, já que nosso território é limitado, passível de desgaste e de uso excludente, isto é, à medida que eu estiver utilizando um pedaço de terra como pastagem, outra pessoa não poderá utilizá-lo para o cultivo.

Ao contrário dos bens tangíveis, as leis de propriedade intelectual referem-se ao direito de uso de bens imateriais, de fonte inesgotável, livres de desgaste e que permitem o uso simultâneo. Uma materialização do conceito de "copo" pode se tornar um bem tangível, escasso, passível de desgaste e de uso excludente, mas a idéia que gerou aquele bem, pode gerar um sem número de modelos e tipos de copos diferentes. Até onde sabemos, não existe um limite da capacidade criadora do ser humano; assim, os produtos da mente vêm de uma fonte inesgotável. Além disso, o fato de uma pessoa estar fazendo uso do conceito "copo" não me impede de utilizar a mesma idéia para criar outro utensílio com as mesmas funcionalidades.

Um exemplo muito usado pelo Movimento do *Software* Livre para ilustrar essas características do bem imaterial e do bem material é a diferença entre a troca de idéias

e de maçãs. Assim, se eu tenho uma maçã e meu vizinho tem outra maçã, quando trocamos nossas maçãs, cada um de nós continua tendo uma maçã. Por outro lado, se cada um de nós tem uma idéia e trocamos essas idéias, ao final do diálogo cada um de nós tem duas idéias.

Tratar a propriedade intelectual como uma propriedade qualquer é um erro, e nem era essa a intenção quando as primeiras regulamentações desse tipo foram criadas. Não se trata apenas de se regulamentar a distribuição de bens escassos na sociedade; as leis de propriedade intelectual foram criadas para estimular a pesquisa, recompensar os inventores e, assim, acelerar o desenvolvimento da sociedade, mas acima de tudo, para garantir o domínio público desses bens depois de um determinado período, para que todos os outros pesquisadores pudessem utilizar aquele conhecimento como matéria prima em suas pesquisas futuras.

É neste contexto que surgem os marcos legais do Movimento de *Software* Livre, que têm como seu maior expoente a *GNU - General Public License* (GPL). Mais do que propor uma forma de trabalho, na construção do conhecimento, esse marco legal propõe uma forma de proteger o conhecimento gerado. No momento em que se coloca em dúvida a possibilidade de se passar novos materiais para o domínio público, com os constantes aumentos dos prazos de proteção, a GPL consolida-se, cada vez mais, como o principal mecanismo, no mundo do *software,* para garantir que aquele conhecimento seja livre e que o conhecimento derivado dele também permaneça assim.

Esse tipo de licenciamento é constantemente rotulado como *copyleft*, em um trocadilho com o *copyright*. Se por um lado o *copyright* significa que não são permitidas cópias e que as obras sob sua proteção têm "todos os direitos reservados" ao detentor da licença, o *copyleft* indica a possibilidade de cópia e manipulação daquela obra, já o seu autor original optou por ter apenas "alguns ou nenhum direito reservado" sobre aquele material.

Como o *copyleft* não existe juridicamente, os autores de *software* que optam por esse tipo de licenciamento registram suas obras como *copyright*, para que ela goze de

todas as proteções previstas pela lei. Assim como seus similares proprietários, os programas livres são vendidos ou distribuídos acompanhados de uma licença de uso, que dirá à pessoa que o comprou ou o baixou da Internet, como poderá utilizar essa ferramenta. No caso da GPL, ela garante ao usuário que obteve o *software* os direitos de usar para qualquer fim, copiar, vender, ou distribuir cópias e alterar o programa original, além de distribuir cópias alteradas do programa.

Mas se a GPL se propõe ser tão libertária, por que não lançar simplesmente o *software* como sendo de domínio público, ou por que não utilizar uma licença já existente, como a BSD *License,* que também é extremamente permissiva?

Sem dúvida, uma das intenções de Richard Stallman ao lançar a GPL era defender os interesses do domínio público. No entanto, havia uma preocupação muito grande por parte dele com a apropriação indevida de seu trabalho. Stallman estava preocupado em não permitir que pessoas ou empresas, que não comungassem da mesma filosofia do Movimento de *Software* Livre, se aproveitassem dos avanços feitos pelo projeto GNU e transformassem qualquer um de seus *softwares* em um programa proprietário.

Tanto o domínio público, como a BSD *License,* utilizada no projeto BSD (*Berkeley Software Distribution*), deixariam brechas para essa possibilidade. Não se questiona, dentro do Movimento de *Software* Livre, a importância da BSD *License;* contudo, além de garantir ao usuário todas as liberdades apresentadas pela GPL, ela também permite que se altere e lance versões proprietárias dos *softwares* BSD.

O toque de genialidade da GPL é justamente usar as leis de *copyright* para obrigar o *software* e as obras dele derivadas a permanecerem livres. Para isso, Stallman implantou uma cláusula em sua licença que serve como uma trava de segurança. Essa "trava" não permite que qualquer produto licenciado sob a GPL seja relicenciado, mesmo que sejam implementadas mudanças significativas no produto original. Esse dispositivo é o que caracteriza a GPL como uma licença viral, pois ela "contamina", não só o *software* original, como todos os trabalhos que forem nele baseados, ou que

contenham parte de seu código.

Em vez de propor o obscurantismo dos códigos como forma de proteger e estimular o desenvolvimento, os *hackers* propõem, a abertura dos códigos e a proteção dos mesmos, para que se transformem em uma espécie de "patrimônio intelectual" em oposição à "propriedade intelectual", servindo como matéria prima para o desenvolvimento e avanço da tecnologia. Abrem, assim, uma frente em defesa do domínio público, cujo principal objetivo e proteger o conhecimento sem aprisioná-lo, garantindo que as gerações futuras tenham acesso à matéria-prima necessária para que possam tocar adiante suas pesquisas.

Em vez de auditar cada um de seus usuários, o *software* livre abre seus códigos de forma que possa ser auditado pelos seus usuários. Esta atitude busca, por meio da transparência, não só maior segurança, estabilidade e velocidade na correção dos erros, já que todos podem verificar os códigos do sistema e corrigir as falhas encontradas, mas também garante que seus usuários não serão tolhidos de nenhuma forma em sua liberdade, por programas de computador, nem terão sua privacidade comprometida por códigos maliciosos embutidos no sistema.

O marco legal do software livre não está voltado apenas para a proteção de direitos individuais do programador, ou da empresa que produziu uma determinada solução, apesar de também protegê-los. Seu foco está voltado à proteção da livre circulação do conhecimento, na construção colaborativa e na defesa do domínio público, propondo, assim, uma oposição entre a antiga idéia de "propriedade intelectual" e um nova noção de "patrimônio intelectual". Se a antiga concepção de propriedade intelectual protegia para que outros não tivessem acesso e não pudessem reproduzir ou usufruir de uma determinada tecnologia sem o pagamento de *royalties*, o patrimônio intelectual deverá proteger de forma a que todos tenham acesso ao conhecimento a fim de recuperar o potencial emancipador da ciência.

Capítulo – 4

Da propriedade ao patrimônio intelectual

À medida que o Movimento de S*oftware* Livre vem ganhando espaço junto a outros setores da sociedade, em grande parte pela popularização das distribuições GNU/Linux, os seus ideais de livre circulação do conhecimento também passam a se popularizar e exercer influência sobre novos segmentos da sociedade. Ações similares às que geraram as comunidades de *software* livre começam a surgir nas artes plásticas, na música, na literatura e no cinema.

Assim como as novas tecnologias da informação facilitaram os processos de cópia de produtos digitais, elas também foram responsáveis pelo barateamento dos custos de produção e distribuição dessas obras. Muitos artistas acharam, nessas novas tecnologias, a viabilidade econômica para a produção, divulgação e distribuição de seu material.

Fortemente influenciados pelos princípios apregoados pelo Movimento de *Software* Livre e pelas possibilidades abertas pelo licenciamento permissivo, cujo maior exemplo é a licença GPL, alguns artistas começaram a propor novas formas de ação para lidar com as limitações das leis de propriedade intelectual e da própria GPL, frente à necessidade de se licenciar outros produtos, além dos *softwares*.

No meio literário, surgiram fenômenos, como o Luther Blissett Porject e posteriormente o projeto Wu Ming. Criado em 1994 por quatro rapazes de Bolonha, Luther Blissett é um pseudônimo multiusuário, por meio do qual o grupo passou a realizar uma "guerrilha midiática" contra a imprensa italiana. Luther Blissett plantou várias histórias falsas em diferentes pontos do país, enganando grandes corporações da mídia italiana e expondo a fragilidade da verdade midiática. À medida que o projeto

ganhou notoriedade, centenas de ativistas, *hackers* e outros operadores culturais passaram a assinar como Luther Blissett.

Durante os anos de 1996 a 1998, quatro membros do grupo Luther Blissett Project - Bolonha redigiram a novela de aventura *Q,* publicada em 1999 pela editora Einaudi e posteriormente traduzida para o inglês, espanhol, alemão, holandês, francês, português (do Brasil), dinamarquês e grego. Em março de 1999, os quatro autores de *Q,* concederam uma entrevista ao diário *La Repubblica* e, quando questionados sobre a razão de não utilizarem seus nomes verdadeiros, os autores afirmam*: Os nossos nomes têm importância mínima e a das nossas histórias individuais é ínfima. Somos o time que escreveu o Q, mas não chegamos a constituir o 0,04% do Luther Blissett Project*. (WU MING FOUNDATION : 2006)

A obra chamou a atenção da mídia pelo número de cópias vendidas, apesar de utilizar uma licença do tipo *copyleft.* Na contracapa do livro, lê-se a inscrição:

> *É consentida a reprodução parcial ou total desta obra bem como a sua distribuição por via telemática para uso pessoal dos leitores, desde que sem fins comerciais, com a condição de que cada copia contenha esta nota.* (BLISSETT – 2002 : 01)

Além das cópias vendidas nas livrarias, também era possível fazer o Download do livro gratuitamente no *site* da editora. O grupo captou o espirito do licenciamento livre proposto pela GPL e o transpôs para sua obra literária.

Em dezembro de 1999, todos os veteranos (que utilizavam o nome Luther Blissett desde 1994) prepararam um suicídio simbólico e encerraram os trabalhos assinados por esta persona. Mesmo assim o pseudônimo continuou a ser adotado por *hackers* e outros ativistas.

Em 2000, mais um escritor se junta aos autores de *Q* e forma-se o grupo Wu

Ming, que em mandarin significa "anônimo". No *site* da *Wu Ming Foundation* (www.wumingfundation.com), eles explicam que *o nome do grupo é entendido como um tributo à dissidência e uma recusa em aceitar o papel do Autor como star. A identidade dos cinco membros do Wu Ming não é segredo, só que consideramos a nossa obra mais importante que as biografias e os rostos individuais.*(WU MING FOUNDATION - 2006). Além do nome coletivo, cada um dos membros possui um nome artístico, composto pelo nome do grupo mais um algarismo escolhido de acordo com a ordem alfabética de seus sobrenomes.

A obra mais conhecida do grupo é *54*, que também foi traduzida para vários idiomas, inclusive o português, e da qual foi extraído um CD, gravado pela banda italiana de rock *yo yo mundi*. Em 2001, é publicado *Havana Glam,* uma obra solo do Wu Ming5; em 2004, são publicados os livros solos de outros dois membros do grupo, *Guerra agli Umani* do Wu Ming2 e *New Thing* do Wu Ming1. Nesse mesmo ano, é exibido nos cinemas o Filme *Con Lentezza,* sobre a Radio Alice e o movimento de 1977 em Bolonha, escrito por Guido Chiesa e Wu Ming. Em comum todas as obras trazem o licenciamento tipo *copyleft*, permitindo a reprodução e distribuição desde que não tivesse fins comerciais.

Em 1998, enquanto Luther Blissett escrevia a sua novela *Q*, na Itália, Lawrence Lessig travava uma batalha judicial nos EUA contra o congresso norte americano que pretendia ampliar, mais uma vez, o período de proteção das obras sob *copyrigth,* por meio da lei Sony Bono. Na época, Lessig era o advogado de Eric Eldred, um programador aposentado que há alguns anos vinha construindo uma biblioteca na Internet, onde publicava livros que estivessem sob domínio público e obras derivadas desses autores. Em 1998, uma parte da obra de Robert Frost passaria para domínio público e Eldred pretendia publicá-la, até que o congresso americano optou por aumentar o periodo de proteção do *copyrigth.* Eldred e Lessig recorreram à suprema corte, para que a decisão do congresso fosse revertida, mas foram derrotados. (LESSIG

- 2006)

Depois desse caso, Lessig se engajou na luta por uma lei de direitos autorais mais justa e criou a Creative Commons, uma empresa sem fins lucrativos, com o objetivo de construir uma camada de *copyright* racional, que se distanciasse dos extremos que atualmente dominam o debate sobre direitos autorais, isto é, todos os direitos reservados, ou nenhum direito reservado. Lessig queria propor uma proteção com alguns direitos reservados.

Para isso, a Creative Commons criou uma série de licenças, a princípio voltadas para trabalhos artísticos e científicos ou que sejam passíveis de proteção pelas leis de direito de autor; em cada uma delas, é possível atribuir um nível de liberdade aos usuários. Tomemos como exemplo um arquivo de música: caso ele seja distribuido sob uma licença *Creative Commons*, o autor poderá especificar cada um dos direitos, que estão sendo previamente liberados ao consumidor.

Ciente das limitações, para se criar uma única opção de licença, a *Creative Commons* procura produzir uma para cada finalidade. Dessa forma, existem licenças específicas para músicas, vídeos, textos e materiais educativos, sendo que em cada uma delas é possível detalhar quais direitos serão concedidos aos usuários. No *site* da *Creative Commons* (www.creativecommons.org.br), é possível gerar automaticamente sua licença a apartir de um questionário simples que deve ser respondido pelo autor.

Quando um produto é licenciado por uma licença *Criative Commons*, são gerados três aquivos referentes a seu produto, uma licença para que possa ser entendida por leigos; assim, qualquer pessoa que acesse um *site* para buscar arquivos sob este tipo de licenciamento poderá entender, de forma clara, quais possibilidades de uso foram dadas pelo autor; o segundo consiste em um arquivo binário, que possibilita que os computadores entendam que aquele é um produto *Creative Commons* e, por último, é gerada uma licença voltada para os advogados.

Uma das idéias por trás do *Creative Commons* é eliminar ao máximo os intermediários. Dessa forma, o autor só precisa ser consultado, caso o uso a ser dado à obra extrapole os direitos concedidos inicialmente por ele.

À medida que ganham força as iniciativas como a dos ativistas do *software* livre e de artistas que optam pelo licenciamento de suas obras sob algum tipo de licenciamento permissivo, aumentam também as pressões dos detentores de *copyrigth*, e de patentes pela ampliação de seus direitos, não só no que se refere ao tempo de proteção, mas também ao escopo do que pode ser protegido.

O TRIPS é hoje uma das maiores ameaças à construção do conhecimento colaborativo, a medida que tenta consolidar um sistema de regulamentação da propriedade intelectual em âmbito mundial, tendo como base a lei norte ameriacana, que é sem dúvida uma das mais rígidas e inapropriadas para o momento histórico em que vivemos. É simplesmente impossível colocar, à disposição das pessoas, tecnologias como máquinas de fotocópias, impressoras de alta resolução, scaners, filmadoras, gravadores de CD e DVD, além de computadores com altíssima capacidade de processamento e armazenamento, e esperar que elas que não utilizem todos os recursos desses equipamentos, por receio de infringirem alguma das leis de *copyrigth*.

Ao mesmo tempo em que a *Sony Music*, processa jovens que compartilham músicas pelas redes P2P, a sua divisão de eletroeletrônicos fornece para esses mesmos jovens computadores de alta performance e *players* portáteis para arquivos MP3. Enquanto uma divisão do conglomerado processa as pessoas que trocam arquivos de música pela internet, outro braço da mesma multinacional fornece as ferramentas necessárias para produzir, trocar, armazenar e reproduzir em qualquer lugar os arquivos tidos como ilegais.

Como resposta a essa pressão que vem sendo imposta pelos donos de *copyrigths*, estão surgindo movimentos políticos organizados em todo o mundo, que culminaram

com a contituição dos partidos piratas. A primeira iniciativa de um partido deste tipo aconteceu na Suécia, onde o partido concorreu às eleições e teve uma votação expressiva, apesar de não ter eleito nenhum deputado. Como principal bandeira o partido defendia a livre troca de arquivos pela Internet, a adoção de *softwares* de código fonte aberto pelo governo, por uma questão de preço e segurança, e por fim, levar ao parlamento europeu a discussão sobre a necessidade de revisão da leis de propriedade intelectual, em uma direção claramente oposta ao TRIPS.

O fenômeno do Partido Pirata, principalmente entre os jovens, fez com que candidatos do Partido Verde da Suécia e de outras agremiações mais à esquerda, que tinham um discurso de combate rigoroso às redes P2P e a circulação de cópias não autorizadas de arquivos pela internet, abrandassem suas posições. Esforços para montar partidos similares sugiram na Alemanha, Áustrália, Áustria, Bélgica, Canadá, Estados Unidos, Espanha, França, Holanda, Itália, Peru, Polônia, Rússia, Reino Unido e África do Sul, além de um Partido Pirata Internacional, que articularia ações conjuntas entre os partidos nacionais.

Falkvinge, o mentor e candidato do Partido Pirata na Suécia, explica que seu partido defende, acima de tudo, a privacidade e a liberdade dos cidadãos.

> *Eu posso te mandar uma música por e-mail. Mas não há diferença entre os bits que formam essa música e os bits que formam a carta que mandei ao meu médico. Então, para defender os direitos autorais, seria preciso monitorar todas as comunicações privadas* (GARATTONI – 2006)

Os militantes do Partido Pirata questionam qual preço deve ser pago para que seja possível sustentar o atual modelo de propriedade intelectual e concluem que, se para isto for necessário abrir mão de toda a comunicação privada na Internet e ter os seus computadores auditados por empresas privadas, então este é um preço muito alto a se pagar.

Porém, o TRIPS e os acordos bilaterais que estão sendo postos em prática pelos EUA estão longe de restringir-se ao combate à cópia não autorizada e à troca de arquivos musicais pela Internet; as multinacionais tentam agora dar um passo adiante, fazendo com que os direitos de patenteabilidade avancem sobre os seres vivos e a biodiversidade dos países em desenvolvimento.

Aprovada em 5 de junho de 1992, a CDB – Convenção sobre Diversidade Biológica – elaborada durante a Conferência das Nações Unidas sobre o Meio Ambiente e Desenvolvimento (CNUMAD) no âmbito da ECO92, colocou pela primeira vez em discussão a necessidade de se proteger os saberes tradicionais, como forma de manutenção da biodiversidade e de modos alternativos de vida indispensáveis para o desenvolvimento sustentável. Em seu Artigo 8, que trata sobre a preservação *in situ,* isto é, a conservação de ecossistemas e hábitats naturais e a manutenção e recuperação de populações viáveis de espécies em seus meios naturais, a CDB, em seu item J, atribui como uma das obrigações dos países membros:

> *Em conformidade com sua legislação nacional, respeitar, preservar e manter o conhecimento, inovações e práticas das comunidades locais e populações indígenas com estilo de vida tradicionais relevantes à conservação e à utilização sustentável da diversidade biológica e incentivar sua mais ampla aplicação com a aprovação e a participação dos detentores desse conhecimento, inovações e práticas; e encorajar a repartição eqüitativa dos benefícios oriundos da utilização desse conhecimento, inovações e práticas;* (MINISTÉRIO DO MEIO AMBIENTE –2002: 12)

Mais de uma década depois, em 17 de outubro 2003, a UNESCO aprovou a Convenção para Salvaguarda do Patrimônio Cultural Imaterial, que apontava na mesma direção. A definição de que tipo de conhecimento é passível de se tornar um patrimônio imaterial pode ser vista no Artigo 2.1, no qual se lê:

> *Se entende por "patrimônio cultural imaterial" os usos, representações, expressões, conhecimentos e técnicas – junto com os instrumentos, objetos, artefatos e espaços culturais que lhes são inerentes – que as comunidades, os grupos, e em alguns casos os indivíduos reconheçam como parte integrante de seu patrimônio cultural. Este patrimônio cultural imaterial que se transmite de geração em geração, é recriado constantemente pelas comunidades e grupos em função de seu entorno, sua interação com a natureza e sua história, incutindo lhes um sentimento de identidade e continuidade e contribuindo assim a promover o respeito da diversidade cultural e da criatividade humana. Aos efeitos da presente convenção, terá-se-á em conta unicamente o patrimônio cultural imaterial que seja compatível com os instrumentos internacionais de direitos humanos existentes e com os imperativos de respeito mútuo entre comunidades, grupos e indivíduos e de desenvolvimento sustentável.* (UNESCO – 2003 : 2)

Mas se é verdade que o processo de contaminação de outras esferas da sociedade pelas noções de racionalização, competência e eficiência científicas fizeram com que as formas alternativas de conhecimento ou interpretações do mundo fossem desprezadas pela ciência ocidental, o que estaria impulsionando esse interesse preservacionista por parte das agências internacionais?

A preocupação em proteger esses conhecimentos tradicionais, não por acaso, chega no mesmo momento em que se busca ampliar a proteção sobre as patentes dos produtos gerados pela nova fronteira da ciência moderna, a biotecnologia, que tem na biodiversidade a matéria prima essencial para seu desenvolvimento e encontra nos conhecimentos tradicionais, técnicas e rituais dos povos indígenas seu principal foco de bioprospecção. Dessa forma, como a própria CDB diz, devemos proteger essa diversidade para *incentivar sua mais ampla aplicação com a aprovação e a participação dos detentores desse conhecimento, inovações e práticas; e encorajar a*

repartição eqüitativa dos benefícios oriundos da utilização desse conhecimento, inovações e práticas. (MINISTÉRIO DO MEIO AMBIENTE – 2002 12)

É claro que "encorajar" não é a mesma coisa que "garantir" a repartição eqüitativa dos benefícios oriundos do conhecimento tradicional. No entanto, o fato de se propor algum tipo de proteção a estes conhecimentos, certamente já é um avanço.

Então, temos aqui um grande dilema: precisamos proteger esses conhecimentos para que seus guardiães não sejam pilhados pelas multinacionais de biotecnologia e, por outro lado, não podemos inviabilizar a livre circulação desse conhecimento, pois dele depende a sobrevivência das comunidades tradicionais e seu estilo de vida.

A proteção desse tipo de conhecimento enfrenta várias dificuldades, sendo que a primeira é seu caráter difuso, resultado da criação coletiva que é transmitida de geração em geração. Estes conhecimentos e tradições são alvo de constantes adaptações e mudanças de acordo com as interações que essas comunidade estabelecem com outros povos e com sua própria história, o que dificulta a definição de titularidade e a identificação da precisa origem geográfica e, conseqüentemente, a justa repartição dos benefícios derivados daquele conhecimento. Outro problema encontrado é que, como vimos, desde 1947, as disputas internacionais sobre proteção da propriedade intelectual passaram a ser discutidas pela OMC, com base no TRIPS que, além de impor um sistema mais rígido de propriedade intelectual, não reconhece a possibilidade de patente de um sistema de conhecimento tradicional, por sua suposta falta de novidade, o que impediria a requisição de uma patente coletiva por qualquer um desses povos.

Vandana Shiva (2001) relata o caso da árvore de nim, cujas propriedades medicinais e bactericidas são amplamente utilizadas na Índia. Considerado por algumas comunidades como uma árvore sagrada, o nim é utilizado na fabricação de biopesticidas e do *dantun,* uma espécie de escova de dentes. A autora lembra que por

séculos comunidades inteiras têm investido dedicação, respeito e conhecimento na propagação, proteção e uso do nim em seus campos, aterros, propriedades rurais e terras comunitárias. No entanto, a partir de 1985, mais de quatorze patentes foram registradas, nos EUA, sobre produtos derivados do nim, sendo principalmente biopesticidas e até mesmo uma pasta de dentes. Em um caso como esse, como deveria ser feita a divisão eqüitativa dos benefícios derivados do conhecimento local, sendo que ele está difundido em quase todo o país?

Para o patenteamento dos produtos derivados do nim, foi atribuído como inovação o processo pelo qual é extraído o princípio ativo. As comunidades indianas que há séculos utilizam o nim como pesticida tiveram os seus conhecimentos saqueados e não receberam um só centavo pelos esforços de "pesquisa e prospecção" que seus ancestrais vinham fazendo e transmitindo de geração em geração.

No atual momento histórico, a biodiversidade deixou de ser um bem local, que há séculos foi indispensável para garantir a sobrevivência de povos dos países subdesenvolvidos do hemisfério sul e passou a ser a biodiversidade mundial, indispensável como matéria prima de empresas multinacionais.

As principais ações propostas pela Convenção para a Salvaguarda do Patrimônio Cultural Imaterial, para a efetiva proteção dos conhecimentos tradicionais estão, na construção de inventários, responsáveis por catalogar esses conhecimentos, e no seu Artigo 13, que dispõe sobre outras medidas de salvaguarda, lê-se:

> *Para assegurar a salvaguarda o desenvolvimento e a valorização do patrimônio cultural imaterial presente em seu território, cada Estado Parte fará todo o possível para:*
>
> *a) adotar uma política geral encaminhada a destacar a função do patrimônio cultural imaterial na sociedade e a integrar sua salvaguarda em programas de planificação;*

b) designar ou criar um ou vários organismos competentes para a salvaguarda do patrimônio cultural imaterial presente em seu território;

c) fomentar estudos científicos, técnico e artísticos, assim como metodologias de investigação. para a salvaguarda eficaz do patrimônio cultural imaterial, e em particular do patrimônio cultural imaterial que se encontre em perigo;

d) adotar as medidas de ordem jurídico, técnico, administrativo e financeiro adequadas para:

i) favorecer a criação e o fortalecimento de instituições de formação em gestão de patrimônio cultural imaterial, assim como a transmissão desse patrimônio nos foros e espaços destinados a sua manifestação e expressão;

ii) garantir o acesso ao patrimônio cultural imaterial, respeitando ao mesmo tempo os usos consuetudinários pelos quais se regem o acesso a determinados aspectos desse patrimônio

iii) criar instituições de documentação sobre o patrimônio cultural imaterial e facilitar o acesso a elas.(UNESCO – 2003 : 5, 6)

Note-se que os itens a, b e c se resumem em conscientizar sobre a importância do patrimônio imaterial e criar órgãos e metodologias de catalogação e preservação desse conhecimento e o item d se detém sobre a importância de facilitar o acesso a esses conhecimentos. Em nenhum momento é dito a quem deverá ser facilitado o acesso. Alguns autores vêem com desconfiança a eficácia desse tipo de banco de dados, pois, ao catalogar todo o patrimônio imaterial de um determinado território, estes arquivos poderiam servir como uma grande vitrine para as multinacionais, indicando onde elas deveriam fazer a sua bioprospecção. Outros autores vêem nesses bancos de dados a

possibilidade concreta de se criar um sistema "*sui generes*" de proteção do conhecimento tradicional e da biodiversidade. Carla Arouca Belas aponta para a seguinte possibilidade:

> *Em termos operacionais, a proposta dos bancos é transformar os conhecimentos tradicionais em segredo industrial, não revelando todo o conhecimento, mas apenas algumas informações, como num catálogo. E, caso haja o interesse, seriam garantidos mecanismos de assessoramento as comunidades para efetivação de negociações e contratos com os interessados. No caso específico dos conhecimentos tradicionais associados à biodiversidade, o CGEN*[7] *poderia, como já o faz, se encarregar da intermediação dos contratos.* (BELAS - 2006)

Mas será que a melhor forma de se proteger o conhecimento tradicional, que foi construído de forma coletiva por esses povos, é o segredo industrial?

No início do capitalismo industrial, vimos a ciência ser convertida em força produtiva e o conhecimento gerado por ela ser aprisionado dentro de instituições privadas, como conseqüência, ela perdeu o seu caráter emancipador e passou a ser mais um mecanismo de controle. Ao transformar o conhecimento tradicional em segredo industrial, podemos estar cometendo o mesmo erro do passado, permitindo que instituições privadas se apropriem do conhecimento coletivo, construído pelas populações tradicionais.

As leis de propriedade intelectual foram, sem dúvida, um dos principais instrumentos para a privatização do conhecimento; e agora, três séculos depois do início desse processo, no momento em que outras tecnologias geraram novas possibilidades, para que a produção do conhecimeto retome seu caráter colaborativo, e a ciência volte a ter lugar no pilar da emancipação, os grupos benefíciados pelo antigo

7 Conselho de Gestão do Patrimônio Genético

modelo de propriedade intelectual lutam para espremer as novas tecnologias e as experiências remanecentes de produção coletiva do conhecimento dentro da forma proposta pelo TRIPS. Mas o fato é que eles já não cabem lá dentro. As novas tecnologias de produção, reprodução e distribuição de bens imateriais, que evoluiram contante e substancialmente nas últimas décadas, vêm dificultando a tal ponto a imposição de tais leis, que em alguns casos elas se tornaram inefetivas. Por outro lado, no mundo todo começam a surgir varios movimentos, grupos organizados e partidos políticos, que questionam a legitimidade de se tratar os bens imateriais da mesma forma que os materiais.

Em vez de se propor novos mecanismos de proteção mais rígidos para os conhecimentos tradicionais, para resguardá-los dos processos predatórios de "biopirataria" que são postos a cabo pelas multinacionais. Devemos propor formas efetivas de abrir o conhecimento que já se encontra aprisionado nessas instituições e que tornam práticas como a "biopirataria" um negócio tão lucrativo.

Alguns indícios dos caminhos que podem ser seguidos surgem em iniciativas, como a do Movimento de *Software* Livre e as licenças *Creative Commons.* No entanto, o que se percebe é que é necessário rever toda a legislação que trata sobre propriedade intelectual. Devemos reavaliar qual seria um prazo socialmente justo para a proteção e, mais que isso, avaliar quais direitos devem ser reservados aos donos do novo *copyrigth.*

Neste momento é necessário nos questionarmos sobre a existencia de uma diferença significativa entre o patrimônio cultural imaterial e a propriedade intelectual. Como vimos no caso do nim indiano, a noção de "inovação", pode ser extremamente questionável. Se ambos são produtos da mente, porque devemos garantir o acesso a um e restringir ao outro?

Quem são os verdadeiros donos das idéias? Quem seria capaz de dizer que gerou

uma única idéia sem recorrer a conhecimentos anteriores?

Todas as nossas idéias e opiniões são construídas a partir de nossas experiências de vida, das conversas com amigos e professores, do que lemos, do que vemos e ouvimos, na TV, no rádio, ou no cinema, além das relações que mantemos com o ambiente em que vivemos. De fato, todas as nossas falas estão impregnadas de citações, mesmo quando não nos damos conta disso.

Esforços como os empenhados pelo Movimento de *Software* Livre tentam resgatar a importância do domínio público das informações e conhecimentos gerados pela ciência e pela tecnologia, para que essa ciência possa continuar avançando, servindo de matéria prima para novas citações, apropriações e transformações desse conhecimento.

Como lembra Silveira:

> *A liberdade para compartilhar e a colaboração em torno do conhecimento são elementos definidores da sociedade em rede. A base imaterial e simbólica destas sociedades exigem novas abordagens que superem aquelas oriundas do controle sobre bens rivais.* (SILVEIRA – 2005 :152)

Se por muito tempo a propriedade intelectual procurou restringir o acesso e privou a ciência de seu caráter emancipador, o patrimônio intelectual deverá garantir o acesso e a colaboração sobre os produtos da mente, para que este quadro possa ser revertido. Resta saber se seremos capazes de utilizar experiencias como a do Movimento de *Software* Livre em outras áreas do conhecimento, consiliando o modelo de produção colaborativa como uma forma de garantir acesso, baratear os custos de pesquisa e dividir de maneira socialmente justa os dividendos desses esforços coletivos. Buscando assim, a construção de um patrimônio intelectual, que proteja e garanta a todos o acesso ao conhecimento, seja ele produzido dentro de laboratórios ou em comunidades tradicionais.

Bibliografia

ALMEIDA, Marco Antônio – A gaiola dos chips – Apontamentos sobre tecnologia, socialbilidade e cultura na sociedade da informação. Revista Em Questão nº 11– Porto Alegre, 2005

APGAUA, Renata – O Linux e a Perspectiva da dádiva - Horizontes Antropológicos, nº21 – Porto Alegre, 2004

AUGUSTO, Maurício Pires – Um estudo sobre as motivações e orientações de usuários e programadores brasileiros de software livre - Rio de Janeiro: UFRJ/COPPEAD, 2003

AUTHIER, Michel – A economia da competência – Revista Margem nº8 - São Paulo – Editora EDUC, 1998

BAUMAN, Zygmunt – Comunidade a busca por segurança no mundo atual – Rio de Janeiro – Jorge Zahar Ed., 2003

BAUMAN, Zygmunt – Modernidade líquida – Rio de Janeiro – Jorge Zahar Ed., 2001

BAXANDALL, Michael – O olhar renascente – Pintura e Experiência Social na Itália da Renascença – Rio de Janeiro – Paz e Terra, 1991

BECK, Ulrich – Liberdade ou capitalismo – São Paulo – Editora UNESP, 2003

BECK, Urich – La Invención de lo político – Para uma teoria de la modernizacion reflexiva – México – Fundo de Cultura Econômica, 1999

BELAS, Carla Arouca - A propriedade Intelectual no âmbito dos direitos difusos: proteção das expressões culturais tradicionais e do conhecimento tradicional associado à biodiversidade – disponível em http://www.museu-

goeldi.br/NPI/download.htm em 09/12/06

BLISSET, Luther – Q – O caçador de hereges – São Paulo – Conrad Editora, 2002

BOHR, Neils – A fisíca atômica e o conhecimento humano – Rio de Janeiro – Contraponto, 1995

BOURDIEU, Pierre – A Economia das Trocas Simbólicas – São Paulo – Editora Perspectiva, 2004

BOURDIEU, Pierre – La Distincion – Criterios e bases sociales del gusto – Madrid -Editora Taurous, 1988

BOURDIEU, Pierre- Meditações Pascalianas – Rio de Janeiro – Bertrand Brasil, 2001

BOYLE, James – Las Ideas Cercadas: El Confinamento y la Desaparicion del Dominio Público – em ¿Un mundo Patenteado? la privatizacion de la vida y el conocimiento – Cordoba – Fundación Via Libre, 2005.

BUAINAIN, Antônio Marcio e CARVALHO, Sérgio M. Paulino de - Propriedade Intelectual em um Mundo Globalizado – trabalho apresentado na Wipo International Conference on Intellectual Property, Trade, Technological Innovation and Competitiveness, Rio de Janeiro, Junho/2000.

BURCKHARDT, Jacob – A cultura do renascimento na Itália – São Paulo – Companhia da Letras, 1991

CASTELLS, Manuel – Asociedade em Rede – A era da informação: Econômia, sociedade e cultura – Volume I – São Paulo – Editora Paz e Terra, 1999

CASTELLS, Manuel – O poder da identidade – A era da informação: Econômia, sociedade e cultura – Volume II – São Paulo – Editora Paz e Terra, 1999

CASTORIADIS, Cornelius – A autonomia em política. O indivíduo privatizado – Revista Margem nº7 - São Paulo – Editora EDUC, 1998

CERVANTES, Silvia Rodríguez - Propiedad intelectual sobre la vida: Estrategias para consolidarla – Disponível em http://www.grain.org/biodiversidad/?id=272

CERVANTES, Silvia Rodríguez – Estrategias cambiantes y combinadas para consolidar la propriedad intelectual sobre la vida y el conocimiento – em ¿Un mundo Patenteado? la privatizacion de la vida y el conocimiento – Cordoba – Fundación Via Libre, 2005.

CRITELLI, Dulce – Martin Heidegger e a essencia da técnica – Revista Margem nº16 - São Paulo – Editora EDUC, 2002

DEBIAN - A FAQ (perguntas freqüentes) do DebianGNU/Linux – 2001, disponível em www.debian.org

DEBIAN – Como reportar bugs no Debian – disponível em www.debian.org

FEYERABEND, Paul K. - Dialogos sobre o conhecimento – São Paulo – Editora Perspectiva, 2001

FOUCAULT, Michel – A arqueologia do saber – Rio de Janeiro – Forense Universitária, 2004

FOUCAULT, Michel – Microfísica do poder – Rio de Janeiro – Edições Graal, 1979

FREE SOFTWARE FOUNDATION – Licenciando software livre – disponível em www.fsf.org em 24/04/05

GARATTONI, Bruno Sayeg - 'Defendemos a privacidade', diz líder – disponível em http://www.link.estadao.com.br/index.cfm?id_conteudo=8616 em 11 de

setembro de 2006

GUIDDENS, Antony – As conseqüências da modernidade – São Paulo – Editora UNESP, 1991

GUIDDENS, Antony; Beck, Ulrich e Lash, Scott – Modernização reflexiva – Política, tradição e estética na ordem social moderna – São Paulo – Editora UNESP, 1997

GUROVITZ, Helio – Linux o fenômeno do software livre – São Paulo – Editora Abril, 2002

HABERMAS, Jürgen – Técnica e Ciência como ideologia – Lisboa – Edições 70, 1994

HALL, Stuart – A identidade cultural na pós-modernidade – Rio de Janeiro – Editora DP&A, 1998

HALL, Stuart – O significado dos novos tempos - – Revista Margem nº7 - São Paulo – Editora EDUC, 1998

IANNI, Octavio – Nacionalismo, regionalismo e globalização – São Paulo – Editora Educ, 1999

INSTITUTO DE ESTUDOS SOCIOECONÔMICOS. Acordo TRIPS: Acordos sobre aspectos dos direitos de rpopriedade intelectual – Brasília – INESC, 2003

JAEGER, Guilherme Pederneiras – Propriedade intelectual – disponível em http://www.iribr.com/cancun/guilherme_pederneiras_jaeger.asp em 10/12/2006

JANDL Junior, Peter - Notas Sobre Sistemas Operacionais – Universidade São Francisco - Faculdade de engenharia. 1999

KEMP, Peter – Da Ética à bioética – Revista Margem nº9 - São Paulo – Editora

EDUC, 1999

KUHN, Thomas – A estrutura das revoluções científicas – São Paulo – Perspectiva, 2005

LESSIG, Lawrence – Cultura livre. Como a mídia usa a tecnologia e a lei para barrar a criação cultural e controlar a criatividade – Trama Universitário, 2006 - disponível em 20/06/2006 em http://www.tramauniversitario.com.br/compartilhe/cultura_livre.jsp

LIMA, Paulo Henrique e SELAIMEN, Graciela – Cúpula mundial sobre a sociedade da informaçã. Um tema de Tod@s – Rio de Janeiro – Rede de Informação para o Terceiro Setor, 2004

MARTIN-BARBERO, Jesus – Arte/Comunicação/Tecnicidade no fim do século – Revista Margem nº8 - São Paulo – Editora EDUC, 1998

MARTIN-BARBERO, Jesus – Dos meios às mediações – Comunicação cultura e hegemonia – Rio de janeiro – Editora UFRJ – 2001

MARTINS, Paulo Roberto – Nanotecnologia, Sociedade e Meio ambiente : 1º seminário internacional – São Paulo – Associação Editorial Humanitas.

MATHEUS, Carlos – Max Scheler e a gênese axiológica do conhecimento – Revista Margem nº16 - São Paulo – Editora EDUC, 2002

MELO Neto, Antônio de Padua e OLIVEIRA, Thiago Tavares Nunes de – Os Limites da propriedade intelectual no ciberespaço: Uma análise do software livre a partir da economia política - trabalho apresentado no Simpósio internacional de propriedade intelectual, informação e ética – Florianópolis, novembro de 2003

MORIN, Edgar – Ciência com consciência – Rio de Janeiro – Bertrand Brasil, 1999

MORIN, Edgar – O método 3 – O conhecimento do conhecimento – Porto Alegre – Sulina, 1999

MORIN, Edgar – O método 4 – As idéias, habitat, vida, costume, organização – Porto Alegre – Sulina, 2002

MUCHAIL, Salma Tanus – Michel Foucault e o dilaceramento do autor – Revista Margem nº16 - São Paulo – Editora EDUC, 2002

NEGROPONTE, Nicholas – A vida digital – São Paulo – Companhia das Letras, 1995

ORGANIZAÇÃO MUNDIAL DE PROPRIEDADE INTELECTUAL - ¿Qué es la Propiedad Intelectual? - disponível em www.ompi.org em 12/04/2005

ORGANIZAÇÃO MUNDIAL DE PROPRIEDADE INTELECTUAL – ? Que es La Propriedad intelectual? - disponíel em www.ompi.int

ORTIZ, Renato – Cultura e Modernidade – São Paulo – Editora Brasiliense, 1991

ORTIZ, Renato – Um outro território – São Paulo – Editora Educ, 1999

PRIMO, Alex Fernando Teixeira. A emergência das comunidades virtuais. In: Intercom 1997 - XX Congresso B r a s i l e i r o de Ciências da Comunicação, 1997, Santos. Anais… Santos, 1997. Disponível em: <http://www.pesquisando.atraves-da.net/comunidades_virtuais.pdf>

RAMOS, Murilo César – Brasil, globalização e redes digitais de banda larga - São Paulo – Editora Educ, 1999

REUTERS - Gorbachev pede ajuda a Bill Gates para pirata russo – disponível em http://tecnologia.terra.com.br/interna/0,,OI1391193-EI4802,00.html em 06/02/2007

RIBEIRO, Lia – Inclusão Digital: com a palavra a sociedade – São Paulo – Plano de Negócios, 2003

RODIN, Josip - Guia do Novo Mantenedor Debian –2002 - disponível em www.debian.org

RODIN, Josip – Guia do novo mantenedor Debian – disponível em www.debian.org

ROSNAY, Joel de – A revolução informacional – Revista Margem nº6 - São Paulo – Editora EDUC, 1997

SANTOS, Boaventura de Souza - A critica da Razão Indolente – Contra o Desperdício da Experiência – São paulo, Editora Cortez, 2002

SANTOS, Laymert Garcia dos – Politizar as novas tecnologias: o impacto sócio-técnico da informação digital e genética – São Paulo – Editora 34, 2003

SCHOLZE Simone H. C. – Acesso ao Patrimônio Genético, Propriedade Intelectual e a Convenção sobre Diversidade Biológica – disponível em http://www.museu-goeldi.br em 09/12/2006

SFEZ, Lucien – Ideologia das novas tecnologias: A Internet e os embaixadores da Comunicação – Revista Margem nº9 - São Paulo – Editora EDUC, 1999

SHIVA, Vandana – Biopirataria - A pilhagem da natureza e do conhecimento – Rio de Janeiro - Editora Vozes, 2001

SHIVA, Vandana – Monoculturas da mente – Perspectivas da biodiversidade e da biotecnologia – São Paulo – Editora Gaia LTDA., 2003

SILVA, Gustavo Noronha - Guia Prático para o Debian GNU/Linux – 2001 – disponível em www.debian.org

SILVEIRA, Sergio Amadeu da – Software livre – A luta pela liberdade do conhecimento – São Paulo – Fundação Perseu Abramo, 2004

SIMON, Imre – A propriedade intelectual na era da internet – 2000 , disponível em www.ime.usp.br/~is/ em 25/11/2006

STALLMAN, Richard - Copyleft: Idealismo Pragmático – 2005 – disponível em www.fsf.org em 26/04/05

STALLMAN, Richard – Copyleft: Idealismo Pragmático – disponível em www.fsf.org em 24/04/05

STALLMAN, Richard – El Projecto GNU – Cordoba - Fundacion Via Libre e Fundacion Heinrich Boll, 2005

STALLMAN, Richard – Está em um nome – diponível em www.fsf.org

STALLMAN, Richard – Lance Software Livre Caso Você Trabalhe em uma Universidade – disponível em www.fsf.org em 24/04/2005

STALLMAN, Richard – Manifesto GNU - Oque é o GNU? GNU não é Unix! – diponível em www.fsf.org

TACHINARDI, Maria Helena – A guerra das patentes – O conflito Braisl x EUA sobre propriedade intelectual – Rio de Janeiro – Paz e Terra,1993

TERRA TECNOLOGIA - Inglesa pode ser presa porque filha usou rede P2P – disponível em www.tecnologia.terra.com.br em 23/06/2005

TORVALDS, Linus e Diamond, David – Só por prazer, Linux os bastidores de sua criação – Rio de Janeiro – Editora Campos, 2001

UNESCO – Convención para la salvaguardia del patrimonio cultural inmaterial – 2003 disponível em www.unesco.org em 25/11/2006

VERSIGNASSI, Alexandre – Folha de São Paulo – Editoria Informática – 26 de abril de 2000

VIRILIO, Paul – Olho por olho ou *crash* da imagens – Revista Margem nº8 - São Paulo – Editora EDUC, 1998

WEBER, Max – Ciência e política duas vocações – São Paulo – Editora Martin Claret, 2004

12 Sites consultados:

Organização	Site
ALSA- Advanced Linux Sound Architecture	www.alsa-project.orgLista
Comunidade Debian	www.debian.org
Comunidade Slackware	www.slackware.org
Debian-BR CDD	http://cdd.debian-br.org/project/
Estúdio Livre	www.estudiolivre.org.br
Free Software Foundation	www.fsf.org
Gnome Foundation	www.gnome.org
Gravadora Trama	www.tramauniversitário.com.br
Internet Archives	
Linux Professional Institute	www.lpi.org.br
lista de discussão do Debian Zine	http://listas.cipsga.org.br/cgi-bin/maiolmanlistinfo/debian-zine
Mozilla Foundation	www.mozilla.org
Museu Paraense Emíllio Goeldi	www.museu-goeldi.br
Organização Mundial de Propriedade Intelectual	www.ompi.org
Projeto Blender	www.belnder.org
Projeto BrOffice	www.broffice.org.br
Projeto Creative Commons	www.creativecommons.org
Projeto Creative Commons Brasil	www.creativecommons.org.br
Projeto GIMP	www.gimp.org
Projeto GNU	www.gnu.org
Projeto Openoffice	www.openoffice.org
PSL Brasil- Projeto de Software Livre Brasil	www.softwarelivre.org

Red-Hat	www.redhat.com
Sourceforce	www.sourceforce.org
Wikipedia – The Free Encyclopedia	www.wikipedia.org

ANEXO – 1

O Manifesto GNU

O Manifesto GNU (que segue abaixo) foi escrito por Richard Stallman no início do Projeto GNU, para pedir por participação e ajuda. Durante os primeiros anos ele sofreu pequenas atualizações para registrar desenvolvimentos, mas agora achamos melhor mante-lo inalterado, já que a maiora das pessoas já viu o manifesto antes.

Desde aquele tempo, nós aprendemos sobre certos mal-entendidos frequentes que uma escolha diferente de palavras poderia ter ajudado a evitar. Notas de rodapé adicionadas em 1993 ajudam a clarear esses pontos.

Para informações atualizadas sobre o software GNU disponível, por favor veja a informação disponível no nosso servidor web, em especial nossa lista de software.

O Que é o GNU? Gnu Não é Unix!

GNU, que significa Gnu Não é Unix, é o nome para um sistema de software completo e compatível com o Unix, que eu estou escrevendo para que possa fornece-lo gratuitamente para todos os que possam utilizá-lo. (1) Vários outros voluntários estão me ajudando. Contribuições de tempo, dinheiro, programas e equipamentos são bastante necessárias.

Até o momento nós temos um editor de textos Emacs com Lisp para a escrita de comandos do editor, um depurador de código-fonte, um gerador de compiladores compatível com o yacc, um linkeditor e em torno de 35 utilitários. Um shell (interpretador de comandos) está quase completo. Um novo compilador C otimizador portável já compilou a si mesmo e deverá ser liberado este ano. Um kernel inicial existe mas muitos recursos ainda são necessários para emular o Unix. Quando o kernel e o compilador estiverem finalizados, será possível distribuir um sistema GNU adequado para o desenvolvimento de novos programas. Nós usaremos o TeX como nosso formatador de textos, mas estamos trabalhando em um nroff. Nós também usaremos o X Window System, que é livre e portável. Depois disso nós adicionaremos um Common Lisp portável, um jogo do Império, uma planilha eletrônica, e centenas de outras coisas, além de documentação on-line. Nós esperamos fornecer, eventualmente, tudo de útil que normalmente vem com um sistema Unix, e ainda mais.

GNU será capaz de rodar programas do Unix, mas não será idêntico ao Unix. Nós faremos todos os aperfeiçoamentos que forem convenientes, baseados em nossa experiência com outros sistemas operacionais. Em particular, nós planejamos adicionar nomes de arquivos longos, números de versão de arquivos, um sistema de arquivos à prova de falhas, auto-geração de nomes de arquivos, talvez, suporte de vídeo independente do terminal, e talvez um sistema de janelas baseado no Lisp atravéz do

qual vários programas Lisp e programas Unix comuns possam compartilhar uma tela. Tanto C quanto Lisp estarão disponíveis como linguagens de programação de sistemas. Nós tentaremos suportar UUCP, MIT Chaosnet, e protocolos da Internet para comunicação.

GNU é inicialmente orientado para máquinas do classe 68000/16000 com memória virtual, porque essas são as máquinas mais fáceis de suportar. O esforço extra par faze-lo rodar em máquinas menores será deixado para alguém que deseje utilizá-lo nelas.

Para evitar uma confusão horrível, por favor pronuncie a letra "G" na palavra "GNU" quando ela for o nome deste projeto.

Por que eu Tenho que Escrever o GNU

Eu acredito que a regra de ouro exige que, se eu gosto de um programa, eu devo compartilha-lo com outras pessoas que gostam dele. Vendedores de Software querem dividir os usuários e conquistá-los, fazendo com que cada usuário concorde em não compartilhar com os outros. Eu me recuso a quebrar a solidariedade com os outros usuários deste modo. Eu não posso, com a consciência limpa, assinar um termo de compromisso de não-divulgação de informações ou um contrato de licensa de software. Por anos eu trabalhei no Laboratório de Inteligência Artificial do MIT para resistir a estas tendências e outras inanimosidades, mas eventualmente elas foram longe demais: eu não podia permanecer em uma instituição onde tais coisas eram feitas a mim contra a minha vontade.

Portanto, de modo que eu possa continuar a usar computadores sem desonra, eu dedicir juntar uma quantidade de software suficiente para que eu possa continuar sem nenhum software que não seja livre. Eu me demiti do Laboratório de IA para impedir que o MIT tenha qualquer desculpa legal para me impedir de fornecer o GNU livremente.

Por que o GNU será Compatível com o Unix

Unix não é o meu sistema ideal, mas ele não é tão ruim. Os recursos essenciais do Unix parecem ser bons recursos, e eu penso que eu posso fornecer o que falta no Unix sem comprometê-lo. E um sistema compatível com o Unix seria conveniente para muitas pessoas adotarem.

Como o GNU Estará Disponível

GNU não está no domínio público. Qualquer um terá permissão para modificar e redistribuir o GNU, mas nenhum distribuidor terá permissão para restringir a sua nova redistribuição. Ou seja, não será permitida nenhuma modificação proprietária (18k characters). Eu quero ter certeza de que todas as versões do GNU permanecerão livres.

Por que Muitos Outros Programadores Desejam Ajudar

Eu encontrei muitos outros programadores que estão excitados quanto ao GNU e querem ajudar.

Muitos programadores estão descontentes quanto à comercialização de software de sistema. Ela pode trazê-los dinheiro, mas ela requer que eles se considerem em conflito com outros programadores de maneira geral em vez de considerá-los como camaradas. O ato fundamental da amizade entre programadores é o compartilhamento de programas; acordos comerciais usados hoje em dia tipicamente proíbem programadores de se tratarem uns aos outros como amigos. O comprador de software tem que escolher entre a amizade ou obeder à lei. Naturalmente, muitos decidem que a amizade é mais importante. Mas aqueles que acreditam na lei frequentemente não se sentem à vontade com nenhuma das escolhas. Eles se tornam cínicos e passam a considerar que a programação é apenas uma maneira de ganhar dinheiro.

Trabalhando com e usando o GNU em vez de programas proprietários, nós

podemos ser hospitaleiros para todos e obedecer a lei. Além disso, GNU serve como um exemplo para inspirar e um chamariz para trazer outros para se juntarem a nós e compartilhar programas. Isto pode nos dar um sentimento de harmonia que é impossível se nós usarmos software que não seja livre. Para aproximadamente metade dos programadores com quem eu falo, esta é uma imporante alegria que dinheiro não pode substituir.

Como Você Pode Contribuir

Eu estou pedindo aos fabricantes de computadores por doações de máquinas e dinheiro. Eu estou pedindo às pessoas por doações de programas e de trabalho.

Uma consequência que você pode esperar se você doar máquinas é que o GNU irá rodar nelas mais cedo. As máquinas devem ser sistemas completos, prontos para uso, e aprovadas para utilização em áreas residenciais, e não devem necessitar de sistemas sofisticados de refrigeração ou energia.

Eu encontrei muitos programadores dispostos a contribuir em tempo parcial com o GNU. Para a maioria dos projetos, este trabalho distribuído em tempo parcial seria bem difícil de coordenar; as partes escritas independentes uma das outras não funcionariam juntas. Mas para a tarefa em particular de substituir o Unix, este problema não existe. Um sistema Unix completo contém centenas de programas utilitários, cada um documentado separadamente. A maiora das especificações de interface são garantidas pela compatibilidade com o Unix. Se cada contribuidor puder escrever um substituto compatível para um único utilitário do Unix, e conseguir que ele trabalhe corretamente no lugar do original em um sistema Unix, então estes utilitários irão funcionar corretamente quando colocados juntos. Mesmo contanto que a Lei de Murphy crie alguns problemas inesperados, juntar estes componentes será um trabalho viável. (O kernel irá necessitar comunicação mais próxima e será trabalhado por um grupo pequeno e coeso.)

Se eu receber doações de dinheiro, eu poderei contratar algumas pessoas em tempo integral ou parcial. O salário não será alto para os padrões da indústria, mas eu estou procurando por pessoas para as quais construir um espírito de comunidade seja tão importante quanto ganhar dinheiro. Eu vejo esta como uma maneira de habilitar pessoas dedicadas a focar as suas energias totalmente no trabalho no GNU, sem que elas necessitem de uma outra maneira de ganhar a vida.

Por que Todos os Usuários de Computadores Serão Beneficiados

Uma vez que o GNU esteja pronto, todos poderão obter um bom software de sistema gratuitamente, assim como o ar. (2)

Isto significa muito mais do que simplesmente que todos economisarão o valor de uma licensa do Unix. Isto significa que muita duplicação de programação de sistemas será evitada. Este esforço poderá ser utilizado em avançar o estado-da-arte.

O código-fonte completo do sistema estará disponível para todos. Como resultado, um usuário que necessite de modificações no sistema será sempre livre para realiza-las ele mesmo, ou para contratar qualquer programador disponível ou empresa para realiza-las. Os usuários não estarão mais à mercê do programador ou empresa que é dono dos fontes e é o único que pode realizar mudanças.

Escolas poderão fornecer um ambiente educacional muito mais produtivo encorajando todos os estudantes a estudar e aperfeiçoar o código do sistema. O Laboratório de Computadores de Harvard tinha como política não instalar nenhum programa se os seus fontes não estivessem disponíveis ao público, e esta posição foi sustentada quando o laboratório se recusou a instalar certos programas. Eu fui bastante inspirado por eles.

Finalmente, o overhead de localizar o dono do software de sistema e o que se pode ou não se pode fazer com ele será aliviado.

Contratos que fazem as pessoas pagarem pelo uso de um programa, incluindo o licensiamento de cópias, sempre trazem um custo tremendo para a sociedade devido aos mecanismos obscuros necessários para se determinar quanto (ou seja, por quais programas) uma pessoa tem que pagar. E somente a polícia do estado tem poder para fazer com que todos obedeçam esses mecanismos. Imagine uma estação espacial onde o ar tem que ser fabricado a um custo muito alto: cobrar cada "respirador" por cada inspiração pode ser justo, mas usar a máscara de gás com o medidor todo dia e toda noite seria intolerável mesmo para os que pudessem pagar a taxa do ar. E presença de câmeras de TV por todo lado para verificar se alguém tirar a máscara é ultrajante. É melhor manter a fábrica de ar com uma taxa por pessoa e eliminar as máscaras.

Copiar todo ou parte de um programa é tão natural para um programador quanto respirar, e tão produtivo quanto. Isto tem que ser livre.

Algumas Objeções Facilmente Refutadas aos Objetivos do GNU

"Ninguém vai utiliza-lo se for gratuito, porque isto significa que não se pode contar com nenhum suporte."

"Você tem que cobrar pelo programa para pagar pelo suporte."

Se as pessoas puderem em vez disso pagar pelo GNU mais pelos serviços em vez de obter o GNU sem o serviço, uma empresa cujo objetivo seja somente fornecer serviços para as pessoas que obteram o GNU gratuitamente será rentável.(3)

Nós temos que diferenciar entre o suporte na forma de verdadeiro trabalho de programação e simples ajuda. O primeiro é algo que ninguém pode realmente contar em receber do vendedor de software. Se o seu problema não é o mesmo de muitas outras pessoas, o vendedor irá ignora-lo.

Se o seu negócio necessita contar com suporte, a única garantia é ter todos os

fontes e ferramentas necessários. Então você pode contratar qualquer pessoa disponível para resolver o seu problema; você não depende de nenhum indivíduo. Com o Unix, o preço dos fontes coloca isto fora de questão para a maioria das empresas. Com GNU seria fácil. Ainda é possível que não haja uma pessoa competente em disponibilidade, mas este problema não será causado por contratos de distribuição. GNU não elimina todos os problemas do mundo, somente alguns deles.

Enquanto isso, o usuário que não sabe nada sobre computadores necessita de ajuda: fazer coisas para eles que eles poderiam facilmente fazer eles mesmos mas eles não sabem como.

Este tipo de serviço poderia ser fornecido por empresas que vendem somente serviços de ajuda e reparos. Se for verdade que os usuários preferem gastar dinheiro e obter o produto com serviço, eles também estarão dispostos à comprar o serviço tendo obtido o produto de graça. As empresas de serviços irão competir em preço e qualidade, enquanto que os usuários não estarão amarrados a nenhuma delas em particular. Enquanto isso, os usuários que não necessitam do serviço poderão usar o programa sem ter que pagar pelo serviço.

"Você não pode atingir muitas pessoas sem propaganda, e você tem que cobrar pelo programa para pagar por isso".

"Não tem sentido anunciar um programa que as pessoas podem pegar de graça".

Existem várias formas de publicidade gratuita ou muito baratas que podem ser usadas para informar os usuários de computadores sobre algo como o GNU. Mas pode ser verdade que nós atingiríamos mais usuários de computadores com propaganda. Se isto for verdade, uma empresa que anuncia o serviço de copiar e enviar GNU por uma taxa será bem-sucedido o suficiente para pagar pelos seus anúncios e mais. Desta

forma, somente os usuários que se beneficiam dos anúncios pagam por eles.

Pelo outro lado, se muitas pessoas copiarem o GNU dos seus amigos, e tais empresas não tiverem sucesso, isto mostra que a propaganda não era realmente necessária para popularizar o GNU. Porque os advogados do mercado livre não deixam o mercado decidir quanto a isso? (4)

"Minha empresa necessita de um sistema operacional proprietário para obter uma vantagem competitiva."

O GNU irá remover o sistema operacional do escopo da competição. Você não será capaz de obter uma vantagem nesta área, mas nenhum dos seus competidores será capaz. Você e eles terão que competir em outras áreas, e se beneficiarão mutuamente nesta área. Se o seu negócio é vender um sistema operacional, você não irá gostar do GNU, mas isto é problema seu. Se o seu negócio é outro, GNU pode poupa-lo de ser forçado para o negócio caro de vender sistemas operacionais.

Eu gostaria de ver o desenvolvimento do GNU suportado por doações de várias empresas e usuários, reduzindo o custo para todos. (5)

"Os programadores não merecem uma recompensa pela sua criatividade?"

Se alguma coisa realmente merece uma recompensa, é a sua contribuição social. Criatividade pode ser uma contribuição social, mas somente na medida em que a sociedade é livre para usufruir dos resultados. Se os programadores merecem ser recompensados por criarem programas inovadores, da mesma forma eles merecem ser punidos se eles restringem o uso destes programas.

"Um programador não deveria poder pedir por uma recompensa pela sua criatividade?"

Não há nada errado em querer pagamento pelo trabalho, ou em procurar

maximizar a renda de uma pessoa, desde que não sejam utilizados meios destrutivos. Mas os meios comuns hoje no campo de software são baseados em destruição.

Extrair dinheiro dos usuários de um programa restringindo o seu uso é destrutivo porque as restrições reduzem a quantidade de vezes e de modos em que o programa pode ser utilizado. Isto reduz a quantidade de bem-estar que a humanidade deriva do programa. Quando há uma escolha deliberada em restringir, as consequências prejudiciais são destruição deliberada.

O motivo pelo qual um bom cidadão não utiliza tais meios destrutivos para se tornar mais rico é porque, se todos fizessem assim, todos nós nos tornaríamos mais pobres pela exploração mútua. Isto é ética Kantiana, ou a Regra de Ouro. Já que eu não gosto das consequências que resultam se todos restringirem a informação, eu tenho que considerar errado para alguém fazer isso. Especificamente, o desejo de ser recompensado pela minha criatividade não justifica privar o mundo em geral de tudo ou parte da minha criatividade.

"Os programadores não irão morrer de fome?"

Eu poderia responder que ninguém é forçado a ser um programador. A maioria de nós não conseguiria nenhum dinheiro pedindo na rua ou fazendo caretas. Mas nós não estamos, como resultado, condenados a passar nossas vidas pedindo na rua, fazendo caretas e passando fome. Nós fazemos outra coisa.

Mas esta é a resposta errada porque ela aceita a afirmação implícita na questão: que sem a propriedade do software, os programadores não tem como receber um centavo. Supõe-se que seja tudo ou nada.

O motivo pelo qual os programadores não irão morrer de fome é que ainda será possível para eles serem pagos para programar; somente não tão bem pagos como o são hoje.

Restringir a cópia não é a única base para negócios com software. Ela é a mais comum porque é a que traz mais dinheiro. Se ela fosse proibida, ou rejeitada pelos consumidores, as empresas de software iriam mover suas bases para outras formas de organização que hoje são utilizadas menos frequentemente. Existem várias formas de se organizar qualquer tipo de negócios.

Provavelmente a programação não será tão lucrativa nas novas bases como ela é agora. Mas este não é um argumento contra a mudança. Não é considerado uma injustiça que caixas de lojas tenham os salários que eles tem hoje. Se com os programdores acontecer o mesmo, também não será uma injustiça. (Na prática eles ainda ganhariam consideravelmente mais do que os caixas.)

"As pessoas não tem o direito de controlar como a sua criatividade é utilizada?"

"Controle sobre o uso das idéias" é na verdade controle sobre as vidas das pessoas; e isto em geral torna as vidas das pessoas mais difícil.

As pessoas que estudaram a questão da propriedade intelectual cuidadosamente (como os advogados) dizem que não existe direito intrínseco sobre a propriedade intelectual. Os tipos de suposta propriedade intelectual que o governo reconhece foram criados por atos específicos de legislação para propósitos específicos.

Por exemplo, o sistema de patentes foi criado para encorajar inventores a divulgarem os detalhes de suas invenções. Seu propósito foi de ajudar à sociedade e não os inventores. Naquela época, o tempo de vida de 17 anos de uma patente era curto comparado com a taxa de avanços no estado-da-arte. Como patentes são um problema somente entre fabricantes, para os quais o custo e o esforço de um contrato de licensa são pequenos se comparados com o custo de se montar uma fábrica, a patente não causou muito prejuízo. Elas não obstruíram a maioria das pessoas que utilizavam produtos patenteados.

A idéia de copyright não existia nos tempos antigos, quando os autores frequentemente copiavam outros autores extensamente em trabalhos denão-ficção. Esta prática era útil, e era a única maneira pela qual o trabalho de muitos autores poderia ter sobrevivido pelo menos em parte. O sistema de copyright foi criado expressamente com o propósito de encorajar a autoria. No domínio para a qual ele foi inventado -- livros que só podiam ser copiados economicamente apenas pela prensa de uma gráfica -- ele causou poucos danos,e não obstruíu a maioria das pessoas que liam os livros.

Todos os direitos de propriedade intelectual são apenas licensas concedidas pela sociedade porque se pensava, corretamente ou não, que a sociedade como um todo se beneficiaria da concessão. Mas, em qualquer situação em particular, temos que perguntar: nós estamos realmente melhor concedendo esta licensa? Que tipo de atos nós estamos autorizando uma pessoa a cometer?

A situação dos programas hoje é bastante diferente daquela dos livros um século atrás. O fato de que o modo mais fácil de copiar um programa é de um vizinho para o outro, o fato de que um programa tem tanto código fonte quanto código objeto que são distintos, e o fato de que um programa é utilizado em vez de lido e apreciado, se combinam para criar uma situação em que uma pessoa que faz valer um copyright está prejudicando a sociedade como um todo tanto material quanto espiritualmente; esta pessoa não deveria fazer isso apesar ou mesmo que a lei permita que ela faça.

"Competição faz com que as coisas sejam feitas melhor."

O paradigma da competição é uma corrida: recompensando o vencedor, nós encorajamos todos a correr mais rápido. Quando o capitalismo realmente funciona deste modo, ele faz um bom trabalho; mas os defensores estão errados em assimir que as coisas sempre funcionam desta forma. Se os corredores se esquecem do porque a recompensa ser oferecida e buscarem vencer, não importa como, eles podem encontrar

outras estratégias -- como, por exemplo, atacar os outros corredores. Se os corredores se envolverem em uma luta corpo-a-corpo, todos eles chegarão mais tarde.

Software proprietário e secreto é o equivalente moral aos corredores em uma luta corpo-a-corpo. É triste dizer, mas o único juiz que nós conseguimos não parece se opor às lutas; ele somente as regula ("para cada 10 metros, você pode disparar um tiro"). Ele na verdade deveria encerrar com as lutas, e penalizar os corredores que tentarem lutar.

"Não irão todos parar de programar sem um incentivo monetário?"

Na verdade, muitas pessoas irão programar sem absolutamente nenhum incentivo monetário. A programação excerce uma fascinação incrível para algumas pessoas, geralmente as pessoas que são melhores nisso. Não há falta de músicos profissionais que se mantém na carreira mesmo quando não há esperança de se ganhar a vida desta forma.

Mas na verdade esta questão, apesar de ser feita frequentemente, não é adequada para a situação. Não se deixará de pagar para os programadores, apenas se pagará menos. Então a questão é, alguém irá programar com um incentivo monetário reduzido? Minha experiência mostra que sim.

Por mais de 10 anos, muitos dos melhores programadores do mundo trabalharam no Laboratório de Inteligência Artificial do MIT por menos dinheiro que eles poderiam receber em qualquer outro lugar. Eles receberam vários tipos de recompensas não-financeiras: fama e reconhecimento, por exemplo. E criatividade também é um entretenimento, uma recompensa em si mesma.

Então a maioria deles saiu quando recebeu uma chance de fazer o mesmo trabalho interessante recebendo bastante dinheiro.

Os fatos mostram que as pessoas irão programador por outros motivos além de ficarem ricas; mas se for dada uma chance para além disso ganharem muito dinheiro,

elas irão aceitar e pedir por isso. Organizações que pagam pouco se comparam fracamente com organizações que pagam bem, mas elas não tem que se realizar seu trabalho de maneira ruim se as organizações que pagam bem forem banidas.

"Nós necessitamos de programadores desesperadamente. Se eles exigem que nós paremos de ajudar nossos semelhantes, nós temos que obedecer."

Você nunca está tão desesperado que você tenha que atender a este tipo de exigência. Lembre-se: milhões para a defesa, mas nenhum centavo como tributo!

"Os programadores tem que ganhar a vida de algum jeito."

Avaliando superficialmente, isto é verdade. Entretanto, existem muitas maneiras pelas quais um programador pode ganhar a vida sem vender o direito de uso de um programa. Este modo é comum hoje porque ele traz aos programadores e aos homens de negócios o máximo em dinheiro, não porque é o único modo de se ganhar a vida. É fácil encontrar outros modos de ganhar a vida se você deseja encontra-los. Eis alguns exemplos.

Um fabricante lançando um novo computador irá pagar pelo porte do sistema operacional para o novo hardware.

A venda de serviços de treinamento, ajuda e manutenção também poderia empregar os programadores.

Pessoas com novas idéias poderiam distribuir programas como freeware, pedindo por doações de usuários satisfeitos, ou vendendo serviços de ajuda [no uso do software]. Eu encontrei pessoas que já trabalham desta forma com sucesso.

Usuários com necessidades parecidas podem formar grupos de usuários, e pagar anuidades. O grupo poderia contratar empresas de programação para escrever programas que os membros do grupo desejariam usar.

Todos os tipos de desenvolvimento podem ser financiados com um Imposto do Software:

Suponha que todos os que compram um computador tenham que pagar X por cento do preço como um imposto do software. O governo daria este dinheiro a uma agência como a NSF para gastar em desenvolvimento de software.

Mas, se um comprador de computadores realizar uma doação para o desenvolvimento de software por conta própria, ele pode abater esta doação do imposto. Ele pode doar para o projeto que ele escolher -- frequentemente escolhido porque ele pretende utilizar os resultados no final. Ele pode ter um crédito por qualquer doação até o total do imposto que ele teria que pagar.

O percentual do imposto poderia ser decidido por voto dos pagadores do imposto, proporcionalmente à quantidade de dinheiro sobre a qual eles serão taxados.

As consequências:

- A comunidade de usuários de computadores suportaria o desenvolvimento de software.
- Esta comunidade decidiria qual nível de suporte é necessário.
- Usuários preocupados com quais projetos a sua parcela é gasta poderiam escolher por eles mesmos.

À longo prazo, tornar os programas livres é um passo adiante na direção do mundo pós-escassez, onde ninguém terá que trabalhar duro somente para ganhar a vida. As pessoas serão livres para se dedicarem às atividades que são agradáveis, como programação, depois de gastar as 10 horas semanais de trabalho obrigatórias em atividades que são necessárias, como legislação, aconselhamento de famílias, reparo de robôs e prospecção de asteróides. Eles não terão necessidade de ganhar a vida

programando.

Nós já reduzimos bastante a quantidade de trabalho que a sociedade como um todo tem que realizar para a sua própria produtividade, mas somente um pouco disso se transformou em lazer para os trabalhadores porque muita atividade não-produtiva é necessária para se acompanhar a atividade produtiva. As principais causas disso são burocracia e medidas bitoladas contra a competição. O software livre irá reduzir grandemente estes desperdícios na área de produção de software. Nós temos que fazer isso, para que os ganhos técnicos em produtividade sejam transformados em menos trabalho para nós.

Notas de rodapé

(1) A escolha de palavras aqui foi descuidada. A intenção era de que ninguém teria que pagar pela *permissão* para usar o sistema GNU. Mas as palavras não deixam isso claro, e as pessoas frequentemente interpretam que elas significam que as cópias do GNU tem sempre que serem distribuídas gratuitamente ou por um valor simbólico. Esta nunca foi a intenção; posteriormente, o manifesto menciona a possibilidade das empresas fornecerem o serviço de distribuição objetivando o lucro. Subsequentemente eu aprendi a distinguir cuidadosamente entre "free" no sentido de liberdade e "free" no sentido de preço. O Software Livre (Free Software) é o software que os usuários tem a liberdade distribuir e modificar. Alguns usuários podem obter cópias sem custo, enquanto que outros podem pagar para receber cópias -- e se a receita ajuda a aperfeiçoar o software, melhor ainda. O mais importante é que qualquer um que tenha uma cópia tenha a liberdade de cooperar com outras pessoas utilizando o software.

(2) Este é outro lugar onde eu falhei em distinguir entre os dois significados de "free". A afirmação como está escrita não é falsa -- você pode obter cópias do GNU

gratuitamente, dos seus amigos ou da Internet. Mas a afirmação sugere a idéia errada.

(3) Várias dessas empresas existem hoje.

(4) A Fundação Para o Software Livre levanta a maior parte dos seus fundos do serviço de distribuição, apesar dela ser uma instituição de caridade em vez de uma empresa. Se *ninguém* escolher obter as cópias comprando da própria FSF, ela será incapaz de realizar o seu trabalho. Mas isto não significa que restrições proprietárias são justificadas para forçar cada usuário a pagar. Se uma pequena fração de todos os usuários fizerem o seu pedido para a FSF, isto será suficiente para manter a FSF operacional. Por isso nós pedimos aos usuários para nos apoiarem desta forma. Você fez a sua parte?

(5) Um grupo de fabricantes de computadores recentemente ofereceu fundos para a manutenção do Compilador C do GNU.

13 ANEXO - 2

Contrato Social Debian

Versão 1.1 ratificada em 26 de Abril de 2004. Substitui a Versão 1.0 ratificada em 5 de Julho de 1997.

O Projeto Debian, produtor do sistema Debian GNU/Linux, criou o Contrato Social Debian. A Definição Debian de Software Livre (DFSG), uma parte do contrato, inicialmente designada como um conjunto de compromissos públicos que nós concordamos em respeitar, foi adotada pela comunidade de software livre como a base para a Definição Open Source.

Contrato Social perante a Comunidade de Software Livre

1. O Debian permanecerá 100% livre

Nós disponibilizamos as definições que usamos para determinar se um software é "livre" no documento intitulado "A Definição Debian de Software Livre (DFSG)". Nós prometemos que o sistema Debian e todos seus componentes serão livres de acordo com essas definições. Nós iremos fornecer suporte às pessoas que desenvolvem ou usam software livre e não-livre no Debian. Nós nunca faremos o sistema depender de um componente não-livre.

2. Nós iremos retribuir à comunidade software livre

Quando escrevermos novos componentes do sistema Debian, nós os licenciaremos de um modo consistente com a Definição Debian de Software Livre. Iremos fazer o melhor sistema que pudermos, de modo que o software livre seja amplamente distribuído e utilizado. Iremos fornecer aos autores originais dos componentes usados em nosso sistema, as correções de bugs, aperfeiçoamentos, solicitações de usuários, etc.

3. Nós não esconderemos problemas

Iremos manter nosso banco de dados de relatório de bugs aberto para a visualização pública todo o tempo. Os relatórios que as pessoas preenchem online ficarão visíveis imediatamente para todos as outras pessoas.

4. Nossas prioridades são nossos usuários e o software livre

Nos guiaremos pelas necessidades de nossos usuários e da comunidade software livre. Colocaremos seus interesses em primeiro lugar nas nossas prioridades. Nós iremos fornecer suporte às necessidades de nossos usuários para que o sistema funcione em diversos tipos de ambientes computacionais. Não faremos objeção a softwares não-livres que têm como objetivo rodar em sistemas Debian, nem tentaremos cobrar taxa alguma às pessoas que criarem ou utilizarem estes softwares. Permitiremos que outras pessoas criem distribuições contendo o sistema Debian e outros softwares, sem cobrar taxa alguma. Como forma de amparar estes objetivos, nós disponibilizaremos um sistema integrado, com materiais de alta qualidade, e sem restrições legais que possam impedir tais usos do mesmo.

5. Programas que não atendem nossos padrões de software livre

Nós reconhecemos que alguns de nossos usuários precisam usar softwares que não atendem à Definição Debian de Software Livre. Criamos as áreas "contrib" e "non-free" em nossos repositórios para estes softwares. Os pacotes contidos nessas áreas não são parte do sistema Debian, embora tenham sido configurados para rodar no Debian. Nós incentivamos os fornecedores de CDs a ler as licenças dos pacotes armazenados nessas áreas, a fim de determinar se podem distribuí-los em seus CDs. Assim, embora softwares não-livres não sejam considerados parte do Debian, nós oferecemos suporte à sua utilização e disponibilizamos infra-estrutura para pacotes não-livres (como nosso sistema de controle de bugs e listas de discussão).

A Definição Debian de Software Livre (DFSG)

1. Redistribuição livre

A licença de um componente Debian não pode restringir nenhuma parte interessada em vendê-lo, ou distribuir o software como parte de uma distribuição agregada de software contendo programas de diversas fontes diferentes. A licença não pode exigir um royalty ou outra taxa por esta venda.

2. Código Fonte

O programa deve incluir código fonte e deve permitir a distribuição em código fonte, bem como em formato compilado.

3. Trabalhos Derivados

A licença deve permitir modificações e trabalhos derivados, e deve permitir que estes sejam distribuídos sob a mesma licença que o trabalho original.

4. Integridade do Código Fonte do Autor

A licença pode restringir o código fonte de ser distribuído de forma modificada _somente_ se a licença permitir a distribuição de "patch files" com o código fonte, com o propósito de modificar o programa em tempo de compilação. A licença deve permitir explicitamente a distribuição de software compilado a partir do código fonte modificado. A licença pode exigir que trabalhos derivados tenham um nome ou número de versão diferente do software original (este é um meio-termo; o grupo Debian encoraja todos os autores a não restringir nenhum arquivo, fonte ou binário, de ser modificado).

5. Não à discriminação contra pessoas ou grupos.

A licença não pode discriminar nenhuma pessoa ou grupo de pessoas

6. Não à discriminação contra Fins de Utilização

A licença não pode restringir ninguém de fazer uso do programa para um fim específico. Por exemplo, ela não pode restringir o programa de ser usado no comércio, ou de ser usado para pesquisa genética.

7. Distribuição de Licença

Os direitos atribuídos ao programa devem aplicar-se a todos aqueles para quem o programa é redistribuído, sem a necessidade de execução de uma licença adicional por aquelas pessoas.

8. A Licença não pode ser específica para o Debian

Os direitos atribuídos ao programa não podem depender do programa ser parte de um sistema Debian. Se o programa for extraído do Debian e usado ou distribuído sem o Debian, dentro dos termos da licença do programa, os mesmos direitos garantidos em conjunto ao sistema Debian deverão ser garantidos àqueles que o utilizam.

9. A Licença não deve contaminar outros softwares.

A licença não poderá colocar restrições em outro software que é distribuído juntamente com o software licenciado. Por exemplo, a licença não pode insistir que todos os outros programas distribuídos na mesma mídia sejam software livre.

10. Licenças Exemplo

As licenças "GPL", "BSD" e "Artistic" são exemplos de licenças que consideramos "livres".

O conceito de declarar nosso "contrato social para a comunidade de software livre" foi sugerido por Ean Schuessler. O rascunho deste documento foi escrito por Bruce Perens, refinado por outros desenvolvedores Debian durante uma conferência

via e-mail que durou um mês em Junho de 1997, e então aceita como uma política pública do Projeto Debian.

Mais tarde, Bruce Perens removeu as referências específicas do Debian da Definição Debian de Software Livre para criar a Definição de Código Aberto.

Outras organizações podem fazer derivações deste documento. Por favor, dê o crédito ao Projeto Debian se você fizer isso.

ANEXO – 3

LICENÇA PÚBLICA GERAL GNU Versão 2, junho de 1991

This is an unofficial translation of the GNU General Public License into Brazilian Portuguese. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL -- only the original English text of the GNU GPL does that. However, we hope that this translation will help Brazilian Portuguese speakers understand the GNU GPL better.

Esta é uma tradução não-oficial da Licença Pública Geral GNU ("GPL GNU") para o português do Brasil. Ela não foi publicada pela Free Software Foundation, e legalmente não afirma os termos de distribuição de software que utiliza a GPL GNU -- apenas o texto original da GPL GNU, em inglês, faz isso. Contudo, esperamos que esta tradução ajude aos que utilizam o português do Brasil a entender melhor a GPL GNU.

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave,

Cambridge, MA 02139, USA

A qualquer pessoa é permitido copiar e distribuir cópias desse documento de licença, desde que sem qualquer alteração.

Introdução

As licenças de muitos software são desenvolvidas para restringir sua liberdade de compartilhá-lo e mudá- lo. Contrária a isso, a Licença Pública Geral GNU pretende

garantir sua liberdade de compartilhar e alterar software livres -- garantindo que o software será livre e gratuito para os seus usuários. Esta Licença Pública Geral aplica-se à maioria dos software da Free Software Foundation e a qualquer outro programa cujo autor decida aplicá-la. (Alguns outros software da FSF são cobertos pela Licença Pública Geral de Bibliotecas, no entanto.) Você pode aplicá-la também aos seus programas.

Quando nos referimos a software livre, estamos nos referindo a liberdade e não a preço. Nossa Licença Pública Geral foi desenvolvida para garantir que você tenha a liberdade de distribuir cópias de software livre (e cobrar por isso, se quiser); que você receba o código-fonte ou tenha acesso a ele, se quiser; que você possa mudar o software ou utilizar partes dele em novos programas livres e gratuitos; e que você saiba que pode fazer tudo isso.

Para proteger seus direitos, precisamos fazer restrições que impeçam a qualquer um negar estes direitos ou solicitar que você deles abdique. Estas restrições traduzem-se em certas responsabilidades para você, se você for distribuir cópias do software ou modificá-lo.

Por exemplo, se você distribuir cópias de um programa, gratuitamente ou por alguma quantia, você tem que fornecer aos recebedores todos os direitos que você possui. Você tem que garantir que eles também recebam ou possam obter o código-fonte. E você tem que mostrar-lhes estes termos para que eles possam conhecer seus direitos.

Nós protegemos seus direitos em dois passos: (1) com copyright do software e (2) com a oferta desta licença, que lhe dá permissão legal para copiar, distribuir e/ou modificar o software.

Além disso, tanto para a proteção do autor quanto a nossa, gostaríamos de certificar-nos que todos entendam que não há qualquer garantia nestes software livres.

Se o software é modificado por alguém mais e passado adiante, queremos que seus recebedores saibam que o que eles obtiveram não é original, de forma que qualquer problema introduzido por terceiros não interfira na reputação do autor original.

Finalmente, qualquer programa é ameaçado constantemente por patentes de software. Queremos evitar o perigo de que distribuidores de software livre obtenham patentes individuais, o que tem o efeito de tornar o programa proprietário. Para prevenir isso, deixamos claro que qualquer patente tem que ser licenciada para uso livre e gratuito por qualquer pessoa, ou então que nem necessite ser licenciada.

Os termos e condições precisas para cópia, distribuição e modificação se encontram abaixo:

LICENÇA PÚBLICA GERAL GNU TERMOS E CONDIÇÕES PARA CÓPIA, DISTRIBUIÇÃO E MODIFICAÇÃO

0. Esta licença se aplica a qualquer programa ou outro trabalho que contenha um aviso colocado pelo detentor dos direitos autorais informando que aquele pode ser distribuído sob as condições desta Licença Pública Geral. O "Programa" abaixo refere-se a qualquer programa ou trabalho, e "trabalho baseado no Programa" significa tanto o Programa em si como quaisquer trabalhos derivados, de acordo com a lei de direitos autorais: isto quer dizer um trabalho que contenha o Programa ou parte dele, tanto originalmente ou com modificações, e/ou tradução para outros idiomas. (Doravante o processo de tradução está incluído sem limites no termo "modificação".) Cada licenciado é mencionado como "você".

Atividades outras que a cópia, a distribuição e modificação não estão cobertas por esta Licença; elas estão fora de seu escopo. O ato de executar o Programa não é restringido e o resultado do Programa é coberto apenas se seu conteúdo contenha trabalhos baseados no Programa (independentemente de terem sido gerados pela execução do Programa). Se isso é verdadeiro depende do que o programa faz.

1. Você pode copiar e distribuir cópias fiéis do código-fonte do Programa da mesma forma que você o recebeu, usando qualquer meio, deste que você conspícua e apropriadamente publique em cada cópia um aviso de direitos autorais e uma declaração de inexistência de garantias; mantenha intactas todos os avisos que se referem a esta Licença e à ausência total de garantias; e forneça a outros recebedores do Programa uma cópia desta Licença, junto com o Programa. Você pode cobrar pelo ato físico de transferir uma cópia e pode, opcionalmente, oferecer garantia em troca de pagamento.

2. Você pode modificar sua cópia ou cópias do Programa, ou qualquer parte dele, assim gerando um trabalho baseadono Programa, e copiar e distribuir essas modificações ou trabalhos sob os temos da seção 1 acima, desde que você também se enquadre em todas estas condições:

a) Você tem que fazer com que os arquivos modificados levem avisos proeminentes afirmando que você alterou os arquivos, incluindo a data de qualquer alteração.

b) Você tem que fazer com que quaisquer trabalhos que você distribua ou publique, e que integralmente ou em partes contenham ou sejam derivados do Programa ou de suas partes, sejam licenciados, integralmente e sem custo algum para quaisquer terceiros, sob os termos desta Licença.

c) Se qualquer programa modificado normalmente lê comandos interativamente quando executados, você tem que fazer com que, quando iniciado tal uso interativo da forma mais simples, seja impresso ou mostrado um anúncio de que não há qualquer garantia (ou então que você fornece a garantia) e que os usuários podem redistribuir o programa sob estas condições, ainda informando os usuários como consultar uma cópia desta Licença. (Exceção: se o Programa em si é interativo mas normalmente não imprime estes tipos de anúncios, seu trabalho baseado no Programa não precisa imprimir um anúncio.)

Estas exigências aplicam-se ao trabalho modificado como um todo. Se seções identificáveis de tal trabalho não são derivadas do Programa, e podem ser razoavelmente consideradas trabalhos independentes e separados por si só, então esta Licença, e seus termos, não se aplicam a estas seções quando você distribui-las como trabalhos em separado. Mas quando você distribuir as mesmas seções como parte de um todo que é trabalho baseado no Programa, a distribuição como um todo tem que se enquadrar nos termos desta Licença, cujas permissões para outros licenciados se estendem ao todo, portanto também para cada e toda parte independente de quem a escreveu.

Desta forma, esta seção não tem a intenção de reclamar direitos os contestar seus direitos sobre o trabalho escrito completamente por você; ao invés disso, a intenção é a de exercitar o direito de controlar a distribuição de trabalhos, derivados ou coletivos, baseados no Programa.

Adicionalmente, a mera adição ao Programa de outro trabalho não baseado no Programa (ou de trabalho baseado no Programa) em um volume de armazenamento ou meio de distribuição não faz o outro trabalho parte do escopo desta Licença.

3. Você pode copiar e distribuir o Programa (ou trabalho baseado nele, conforme descrito na Seção 2) em código-objeto ou em forma executável sob os termos das Seções 1 e 2 acima, desde que você faça um dos seguintes:

a) O acompanhe com o código-fonte completo e em forma acessível por máquinas, que tem que ser distribuído sob os termos das Seções 1 e 2 acima e em meio normalmente utilizado para o intercâmbio de software; ou,

b) O acompanhe com uma oferta escrita, válida por pelo menos três anos, de fornecer a qualquer um, com um custo não superior ao custo de distribuição física do material, uma cópia do código-fonte completo e em forma acessível por máquinas, que tem que ser distribuído sob os termos das Seções 1 e 2 acima e em meio normalmente utilizado para o intercâmbio de software; ou,

c) O acompanhe com a informação que você recebeu em relação à oferta de distribuição do código- fonte correspondente. (Esta alternativa é permitida somente em distribuição não comerciais, e apenas se você recebeu o programa em forma de código-objeto ou executável, com oferta de acordo com a Subseção b acima.)

O código-fonte de um trabalho corresponde à forma de trabalho preferida para se fazer modificações. Para um trabalho em forma executável, o código-fonte completo significa todo o código-fonte de todos os módulos que ele contém, mais quaisquer arquivos de definição de "interface", mais os "scripts" utilizados para se controlar a compilação e a instalação do executável. Contudo, como exceção especial, o código-

fonte distribuído não precisa incluir qualquer componente normalmente distribuído (tanto em forma original quanto binária) com os maiores componentes (o compilador, o "kernel" etc.) do sistema operacional sob o qual o executável funciona, a menos que o componente em si acompanhe o executável.

Se a distribuição do executável ou código-objeto é feita através da oferta de acesso a cópias de algum lugar, então ofertar o acesso equivalente a cópia, do mesmo lugar, do código-fonte equivale à distribuição do código-fonte, mesmo que terceiros não sejam compelidos a copiar o código-fonte com o código- objeto.

4. Você não pode copiar, modificar, sub-licenciar ou distribuir o Programa, exceto de acordo com as condições expressas nesta Licença. Qualquer outra tentativa de cópia, modificação, sub-licenciamento ou distribuição do Programa não é valida, e cancelará automaticamente os direitos que lhe foram fornecidos por esta Licença. No entanto, terceiros que de você receberam cópias ou direitos, fornecidos sob os termos desta Licença, não terão suas licenças terminadas, desde que permaneçam em total concordância com ela.

5. Você não é obrigado a aceitar esta Licença já que não a assinou. No entanto, nada mais o dará permissão para modificar ou distribuir o Programa ou trabalhos derivados deste. Estas ações são proibidas por lei, caso você não aceite esta Licença. Desta forma, ao modificar ou distribuir o Programa (ou qualquer trabalho derivado do Programa), você estará indicando sua total aceitação desta Licença para fazê-los, e todos os seus termos e condições para copiar, distribuir ou modificar o Programa, ou trabalhos baseados nele.

6. Cada vez que você redistribuir o Programa (ou qualquer trabalho baseado

nele), os recebedores adquirirão automaticamente do licenciador original uma licença para copiar, distribuir ou modificar o Programa, sujeitos a estes termos e condições. Você não poderá impor aos recebedores qualquer outra restrição ao exercício dos direitos então adquiridos. Você não é responsável em garantir a concordância de terceiros a esta Licença.

7. Se, em conseqüência de decisões judiciais ou alegações de infringimento de patentes ou quaisquer outras razões (não limitadas a assuntos relacionados a patentes), condições forem impostas a você (por ordem judicial, acordos ou outras formas) e que contradigam as condições desta Licença, elas não o livram das condições desta Licença. Se você não puder distribuir de forma a satisfazer simultaneamente suas obrigações para com esta Licença e para com as outras obrigações pertinentes, então como conseqüência você não poderá distribuir o Programa. Por exemplo, se uma licença de patente não permitirá a redistribuição, livre de "royalties", do Programa, por todos aqueles que receberem cópias direta ou indiretamente de você, então a única forma de você satisfazer a ela e a esta Licença seria a de desistir completamente de distribuir o Programa.

Se qualquer parte desta seção for considerada inválida ou não aplicável em qualquer circunstância particular, o restante da seção se aplica, e a seção como um todo se aplica em outras circunstâncias. O propósito desta seção não é o de induzi-lo a infringir quaisquer patentes ou reivindicação de direitos de propriedade outros, ou a contestar a validade de quaisquer dessas reivindicações; esta seção tem como único propósito proteger a integridade dos sistemas de distribuição de software livres, o que é implementado pela prática de licenças públicas. Várias pessoas têm contribuído generosamente e em grande escala para os software distribuídos usando este sistema, na certeza de que sua aplicação é feita de forma consistente; fica a critério do autor/doador decidir se ele ou ela está disposto a distribuir software utilizando outro sistema, e um licenciado não pode impor qualquer escolha.

Esta seção destina-se a tornar bastante claro o que se acredita ser conseqüência do restante desta Licença.

8. Se a distribuição e/ou uso do Programa são restringidos em certos países por patentes ou direitos autorais, o detentor dos direitos autorais original, e que colocou o Programa sob esta Licença, pode incluir uma limitação geográfica de distribuição, excluindo aqueles países de forma a tornar a distribuição permitida apenas naqueles ou entre aqueles países então não excluídos. Nestes casos, esta Licença incorpora a limitação como se a mesma constasse escrita nesta Licença.

9. A Free Software Foundation pode publicar versões revisadas e/ou novas da Licença Pública Geral de tempos em tempos. Estas novas versões serão similares em espírito à versão atual, mas podem diferir em detalhes que resolvem novos problemas ou situações.

A cada versão é dada um número distinto. Se o Programa especifica um número de versão específico desta Licença que se aplica a ele e a "qualquer nova versão", você tem a opção de aceitar os termos e

condições daquela versão ou de qualquer outra versão publicada pela Free Software Foundation. Se o programa não especifica um número de versão desta Licença, você pode escolher qualquer versão já publicada pela Free Software Foundation.

10. Se você pretende incorporar partes do Programa em outros programas livres cujas condições de distribuição são diferentes, escreva ao autor e solicite permissão. Para o software que a Free Software Foundation detém direitos autorais, escreva à Free Software Foundation; às vezes nós permitimos exceções a este caso. Nossa decisão será guiada pelos dois objetivos de preservar a condição de liberdade de todas as derivações do nosso software livre, e de promover o compartilhamento e reutilização de software em aspectos gerais.

AUSÊNCIA DE GARANTIAS

11. UMA VEZ QUE O PROGRAMA É LICENCIADO SEM ÔNUS, NÃO HÁ QUALQUER GARANTIA PARA O PROGRAMA, NA EXTENSÃO PERMITIDA PELAS LEIS APLICÁVEIS. EXCETO QUANDO EXPRESSADO DE FORMA ESCRITA, OS DETENTORES DOS DIREITOS AUTORAIS E/OU TERCEIROS DISPONIBILIZAM O PROGRAMA "NO ESTADO", SEM QUALQUER TIPO DE GARANTIAS, EXPRESSAS OU IMPLÍCITAS, INCLUINDO, MAS NÃO LIMITADO A, AS GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E AS DE ADEQUAÇÃO A QUALQUER PROPÓSITO. O RISCO TOTAL COM A QUALIDADE E DESEMPENHO DO PROGRAMA É SEU. SE O PROGRAMA SE MOSTRAR DEFEITUOSO, VOCÊ ASSUME OS CUSTOS DE TODAS AS MANUTENÇÕES, REPAROS E CORREÇÕES.

12. EM NENHUMA OCASIÃO, A MENOS QUE EXIGIDO PELAS LEIS APLICÁVEIS OU ACORDO ESCRITO, OS DETENTORES DOS DIREITOS AUTORAIS, OU QUALQUER OUTRA PARTE QUE POSSA MODIFICAR E/OU REDISTRIBUIR O PROGRAMA CONFORME PERMITIDO ACIMA, SERÃO RESPONSABILIZADOS POR VOCÊ POR DANOS, INCLUINDO QUALQUER DANO EM GERAL, ESPECIAL, ACIDENTAL OU CONSEQÜENTE, RESULTANTES DO USO OU INCAPACIDADE DE USO DO PROGRAMA (INCLUINDO, MAS NÃO LIMITADO A, A PERDA DE DADOS OU DADOS TORNADOS INCORRETOS, OU PERDAS SOFRIDAS POR VOCÊ OU POR OUTRAS PARTES, OU FALHAS DO PROGRAMA AO OPERAR COM QUALQUER OUTRO PROGRAMA), MESMO QUE TAL DETENTOR OU PARTE

TENHAM SIDO AVISADOS DA POSSIBILIDADE DE TAIS DANOS.

FIM DOS TERMOS E CONDIÇÕES

Como Aplicar Estes Termos aos Seus Novos Programas

Se você desenvolver um novo programa, e quer que ele seja utilizado amplamente pelo público, a melhor forma de alcançar este objetivo é torná-lo software livre que qualquer um pode redistribuir e alterar, sob estes termos.

Para isso, anexe os seguintes avisos ao programa. É mais seguro anexá-los logo no início de cada arquivo-fonte para reforçarem mais efetivamente a inexistência de garantias; e cada arquivo deve possuir pelo menos a linha de "copyright" e uma indicação de onde o texto completo se encontra.

<uma linha que forneça o nome do programa e uma idéia do que ele faz.>

Copyright (C) <ano> <nome do autor>

Este programa é software livre; você pode redistribuí-lo e/ou modificá-lo sob os termos da Licença Pública Geral GNU, conforme publicada pela Free Software Foundation; tanto a versão 2 da Licença como (a seu critério) qualquer versão mais nova.

Este programa é distribuído na expectativa de ser útil, mas SEM QUALQUER GARANTIA; sem mesmo a garantia implícita de COMERCIALIZAÇÃO ou de ADEQUAÇÃO A QUALQUER PROPÓSITO EM PARTICULAR. Consulte a Licença Pública Geral GNU para obter mais detalhes.

Você deve ter recebido uma cópia da Licença Pública Geral GNU junto com este programa; se não, escreva para a Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA.

Inclua também informações sobre como contactá-lo eletronicamente e por carta.

Se o programa é interativo, faça-o mostrar um aviso breve como este, ao iniciar um modo interativo:

Gnomovision versão 69, Copyright (C) ano nome do autor

O Gnomovision não possui QUALQUER GARANTIA; para obter mais detalhes digite `show w'. Ele é software livre e você está convidado a redistribui-lo sob certas condições; digite `show c' para obter detalhes.

Os comandos hipotéticos `show w' e `show c' devem mostrar as partes apropriadas da Licença Pública Geral. Claro, os comandos que você usar podem ser ativados de outra forma que `show w' e `show c'; eles podem até ser cliques do mouse ou itens de um menu -- o que melhor se adequar ao programa.

Você também deve obter do seu empregador (se você trabalha como programador) ou escola, se houver, uma "declaração de ausência de direitos autorais" sobre o programa, se necessário. Aqui está um exemplo; altere os nomes:

Yoyodyne, Inc., aqui declara a ausência de quaisquer direitos

autorais sobre o programa `Gnomovision' (que executa interpretações

em compiladores) escrito por James Hacker.

<assinatura de Ty Coon>, 1o. de abril de 1989

Ty Con, Vice-presidente

Esta Licença Pública Geral não permite incorporar seu programa em programas

proprietários. Se seu programa é uma biblioteca de sub-rotinas, você deve considerar mais útil permitir ligar aplicações proprietárias com a biblioteca. Se isto é o que você deseja, use a Licença Pública Geral de Bibliotecas GNU, ao invés desta Licença.

14 ANEXO – 4

O Manifesto Debian

Escrito por Ian A. Murdock, revisado em 01/06/94

A.1 O que é o Debian Linux?

Debian Linux é um novo tipo de distribuição Linux. Ao invés de ser desenvolvido por uma ou um grupo isolado de pessoas, como outras distribuições de Linux foram, o Debian está sendo desenvolvida abertamente, no espírito do Linux e da GNU. O objetivo principal do Projeto Debian é criar uma distribuição que viva acima do nome Linux. A Debian está sendo feito cuidadosamente e conscientemente, e será mantido da mesma forma.

É também uma tentativa de criar uma distribuição não-comercial, que será capaz de competir eficientemente no mercado comercial. Será, eventualmente, distribuída pela Free Software Foundation em CD-ROM, e a Associação Debian GNU/Linux oferecerá a distribuição em disquetes e fitas, juntamente com manuais impressos, suporte técnico e outros itens essenciais para o usuário final. O citado acima estará disponível por pouco mais que o custo, e o resto será aplicado no desenvolvimento do software livre para todos os usuários. Tal distribuição é essencial ao sucesso do sistema operacional GNU/Linux no mercado comercial, e deve ser feito através de organizações numa posição em que se possa avançar e defender o software

livre sem visar lucros ou retornos.

A.2 Por que o Debian está sendo construído?

Distribuições são essenciais ao futuro do Linux. Especialmente, se elas eliminam a necessidade do usuário localizar, copiar, compilar, instalar e integrar um enorme número de ferramentas essenciais para construir um sistema Linux. Porém, o trabalho de construção do sistema é associado ao criador da distribuição, cujo trabalho pode ser compartilhado com milhares de outros usuários. Quase todos os usuários de Linux terão seu primeiro contato com esse sistema através de uma distribuição, e a maioria desses usuários continuará usando uma distribuição por questão de conveniência, depois que eles estejam familiarizados com o sistema operacional. Desta maneira, as distribuições representam um papel realmente importante.

Apesar da óbvia importância, as distribuições têm chamado a atenção de desenvolvedores. Há uma razão simples para isso: elas não são simples nem \u2019legais\u2019 de construir e requerem uma grande quantidade de esforço e tempo de seu criador para que ela mantenha-se livre de erros e sempre atualizada. Uma coisa é criar um sistema do \u2019nada\u2019. Outra coisa é ter certeza que o sistema é fácil dos outros instalarem, que funcionará com uma larga variedade de configurações de hardware, que conterá programas que serão úteis aos outros, e que será atualizado quando seus componentes são melhorados.

Muitas distribuições começaram como sistemas muito bons, mas com o passar do tempo, a manutenção da distribuição recebe uma atenção secundária. Um exemplo é a Softlanding Linux System (mais conhecida como SLS). É possivelmente a distribuição que possui maior número de problemas e de pior manutenção, mas,

infelizmente, pode ser também a mais popular. É, com certeza, a distribuição que atrai mais atenção dos "distribuidores comerciais" de Linux que se aproveitam da crescente popularidade desse sistema.

Esta é realmente uma combinação ruim, pois a maioria das pessoas que obtém o Linux desses "distribuidores" recebe uma distribuição cheia de defeitos e muito mal administrada. Como se isso não fosse suficiente, esses "distribuidores" têm uma tendência a promover "funções" de seus produtos que não são funcionais ou extremamente instáveis. Some isso ao fato de que os compradores irão, logicamente, esperar do produto todas as suas funções funcionando perfeitamente e que alguns acreditam que ele seja um sistema operacional comercial (também há uma tendência a não mencionar que o Linux é livre e que é distribuído sob a Licença Pública Geral GNU). Finalizando, esses "distribuidores" estão atualmente ganhando bastante dinheiro para manter anúncios enormes em revistas; é o clássico exemplo de comportamento inaceitável sendo recompensado por aqueles que não sabem muito. Definitivamente algo precisa ser feito para remediar a situação.

A.3 Como o Debian tentará pôr fim a esses problemas?

O processo de planejamento do Debian é aberto para que se tenha certeza que o sistema é da mais alta qualidade e que ele reflete as necessidades da comunidade de usuários. Por envolver muitas pessoas que têm diferentes habilidades e realidades, o Debian é capaz de ser desenvolvido de maneira modular. Seus componentes são de alta qualidade, pois, aqueles que têm mais experiência em uma certa área, têm a oportunidade de construir ou manter os componentes individuais do Debian pertinentes àquela área. Envolver outras pessoas também assegura que muitas sugestões muito úteis podem ser dadas e assim melhorar o sistema como um todo

durante o seu desenvolvimento; desta maneira, uma distribuição é criada baseando-se principalmente nas necessidades dos usuários, ao invés das necessidades de seu construtor. É muito difícil para uma única pessoa ou um pequeno grupo de pessoas prever essas necessidades e desejos sem ter contato direto com outras pessoas.

O Debian GNU/Linux também será distribuído em mídia física pela Free Software Foundation e pela Debian GNU/Linux . Isso torna disponível o Debian aos usuários que não têm acesso ao servidor FTP na Internet e também gera produtos e serviços, como manuais impressos e suporte técnico disponível para todos os usuários do sistema. Dessa maneira, o Debian pode ser usado pelo maior número possível de pessoas e corporações, a meta será prover um produto de primeira qualidade, não obter lucros ou retornos, e as melhorias providas ao software serão úteis ao usuário, tendo ele pago ou não.

A Free Software Foundation representa uma peça importantíssima ao futuro do Debian. Pelo simples fato de distribuí-lo, uma mensagem estará sendo enviada ao mundo dizendo que o Linux não é um produto comercial e nunca deverá ser, mas não significa que o Linux não será capaz de competir com produtos comerciais. Para aqueles que discordam disso, desafio a imaginar o sucesso do GNU Emacs edo GCC, que não são produtos comerciais, porém produziram um grande impacto no mercado comercial, apesar desse fato.

Chegou a hora de concentrar-se no futuro do Linux mais do que no destrutivo objetivo de enriquecer uma pessoa às custas da comunidade Linux inteira e de seu futuro. O desenvolvimento e a distribuição do Debian podem não ser a solução para os problemas que eu salientei no Manifesto, mas espero que atraia atenção suficiente para esses problemas, e para que eles sejam resolvidos.