Filosofia GNU

Site:	Moodle da Comunidade
Curso:	Filosofia GNU
Livro:	Filosofia GNU
Impresso por:	Arnaldo Mayr
Data:	terça, 9 fevereiro 2010, 18:48

Sumário

1 Processo Histórico

1.1 Como era o mundo nos anos 70 ?

Em geral, quando falamos em software livre, as pessoas tendem automaticamente a pensarem que se existe algo que é livre, então, comparativamente, tem uma outra "coisa" que é "presa", "amarrada", "acorrentada", sendo isso uma compreensão verdadeira. No decorrer deste curso você vai poder observar que estas correntes que prendem o software estão diretamente ligadas a conceitos de Propriedade Intelectual ou se preferir, as licenças.

Nossa história começa nos anos 70, quando era muito comum para um programador trocar suas experiências de programação com outros parceiros. Quando alguém desenvolvia uma rotina para calcular um intervalo de datas (por exemplo) e um outro programador tinha conhecimento que aquela rotina já estava produzida por alguém, ele tomava a iniciativa de pedi-la ou obtê-la junto ao criador do programa, aproveitando o código e inserindo noutros programas.

Logo, podemos afirmar que no início do mundo da programação os programas de computadores já eram livres, e que os técnicos da área já compartilhavam entre si as soluções tecnológicas, aproveitando o trabalho individual e transformando-o em uma solução coletiva.

É importante observar que durante este mesmo período dos anos 70, as escolas de uma forma geral tinham objetivos mais nobres, voltados ao ensino participativo, como por exemplo: se ao chegar a hora do lanche, uma criança não tinha levado sua merenda, a professora pedia a outra criança que compartilhasse, que dividisse o lanche com o coleguinha. Essa iniciativa da professora gerava nas crianças um aprendizado sobre relacionamento humano com a prática do compartilhamento, do apoio àqueles que tinham necessidade, e muitas outras questões que serviam para o desenvolvimento da sociedade.

1.2 O que mudou nos anos 80 ?

Nos idos dos anos 80 surgiram licenciamentos de software que até então não eram utilizados, impedindo a cooperação dos programadores. Fatos como limites de usuários por produtos, limitação no número de processadores utilizados, e inúmeras outras alternativas se propagavam nas licenças de software.

Até os anos 70 ainda era comum adquirir-se os sistemas computacionais junto com os programas fontes. Na década posterior, os fontes já não faziam mais parte das negociações, os sistemas eram apenas vendidos com uma concessão de uso, as limitações quanto a utilização dos programas eram restritas e diferiam de toda e qualquer relação anteriormente exercida no mercado consumidor.

Nunca em toda historia da humanidade houve uma restrição tão grande. Comparando a outros produtos, imagine comprar uma televisão que poderia apenas ser vista por um limitado número de pessoas, e quando sua família crescesse, seria necessário adquirir mais uma licença, pagando um adicional. Além disso, tome ainda como exemplo que a cada dois anos você teria de adquirir um novo televisor para poder assistir a programações mais atuais, pois a TV não teria a capacidade de "entender" a nova grade de programação, que só seria entendida pela versão mais nova.

É notável ainda que nos anos 80 (especificamente nos Estados Unidos) as escolas passam a impedir os alunos de distribuírem programas de computador através de CDs ou Disquetes, ensinando às crianças uma obediência e respeito aos conceitos de propriedade intelectual, como se isto fosse natural a todos, diferenciando o anterior ensino cidadão dos anos 70, por uma educação a conceitos não naturais a nossa sociedade.

1.3 Construindo o movimento do "software livre"

rms.jpg

Richard Stallman (RMS) era uma das pessoas que trabalhavam no MIT e que havia vivido a fase da liberdade entre os programadores. RMS tinha um sentimento muito forte com relação à liberdade e à comunidade solidária que ele fazia parte.

A história do software livre começa com uma impressora laser que o MIT havia ganho, e que substituiria uma impressora matricial que era utilizada há vários anos pelos técnicos.

Como a impressora antiga fazia parte da história em que a liberdade era comum no uso da tecnologia, o driver utilizado na matricial já continha inserções no código produzidas pelos técnicos do MIT, que permitia informações tais como o momento em que iniciava ou terminava um trabalho de impressão, erros no processo, entre outros. No entanto, a impressora laser não continha essas facilidades, o que causava um grande transtorno.

RMS toma a iniciativa e procura o representante da impressora para tentar negociar com ele a inserção das mesmas facilidades da impressorinha matricial. A resposta que ele ouviu foi a mesma que qualquer pessoa ouviria ao tentar negociar uma alteração com uma empresa de software proprietário, que seria algo mais ou menos como: *nós somos os donos do software e se você quiser alguma alteração, peça-nos, que nós vamos avaliar e implementar se acharmos adequado*.

RMS ainda tentou convencer o representante, que ao final de um longo debate ofertou um contrato para que os programadores assinassem, basicamente constituído por cláusulas de não revelação, ou seja, *Nondisclosure agreement*, uma maneira de conceder acesso a um código, impedindo a divulgação do que fosse visto, conhecido.

Os contratos de não revelação já eram comuns desde a época dos PDPs (*Programmable Data Processor,* uma linha de computadores que foi produzida pela *Digital Equipment Corporation -DEC*- entre 1960 e 1972, indo do modelo PDP-1 ao PDP-16. O modelo mais popular foi o PDP-11 de 1970), e tinha dois objetivos básicos:

1) O mundo da informática (tanto do ponto de vista do software quanto de hardware) era uma novidade, e as empresas de tecnologia da época tentavam estabelecer mecanismos que garantissem o mercado;

2) A guerra fria era um motivo forte, e havia o receio dos americanos de que os seus adversários pudessem utilizar os computadores para fazerem cálculos vetoriais, e com isso,

lançassem uma bomba atômica sobre os EUA.

Finalmente, a proposta do representante da impressora foi considerada por RMS como uma agressão a liberdade. Afinal, como seria possível assinar um acordo antecipado em que você se negaria a ajudar qualquer pessoa, seja ela quem fosse. Além disso, o trabalho do MIT seria incorporado a um produto que continuaria em poder do fabricante, e outras pessoas não teriam oportunidade de aproveitar-se das facilidades que seriam incorporadas pelos técnicos.

Foi a partir dai que Richard Stallman se deu conta que não havia sentido trabalhar em algo proprietário, e que aquele era um momento importante onde as pessoas deviam ter direito ao livre acesso da tecnologia, e para tanto, deu meia volta e começou ainda em 1983 o movimento Free Software, tendo em 1984 fundado a Free Software Foundation.

Inicialmente Richard Stallman criou o GCC, que é o compilador livre de C, o editor de textos EMACS, permitindo assim que vários programadores ao redor do mundo começassem a contribuir na construção de um sistema operacional e de um kernel livre.

O primeiro email enviado por Richard Stallman foi disponibilizado para as listas net.unix-wizards e net.usoft, contendo as idéias básicas do que ele pretendia fazer. Isto aconteceu em 27/09/1983

2 Principais diferenças

2.1 O que é Software Proprietário ?

O software proprietário é anti-social, não é ético, pois divide o público e mantêm os usuários desamparados e dependentes.

A frase acima é o melhor resumo do que é um software proprietário. Para entender melhor, vamos separar partes do texto e explicitá-lo com exemplos, fazendo assim com que a compreensão seja mais simples.

1 - é anti-social e não é ético => Diz-se que o software proprietário é anti-social pelo fato da obrigação de recompra de licenças a cada dois anos. Imagine uma empresa como o Datasus, que possui cerca de 120 mil computadores espalhados por todo Brasil. Para que cada computador destes funcione, é necessário que o mesmo contenha um sistema operacional e pelo menos uma suite de escritório, já que a grande maioria das pessoas utiliza o computador para escrever textos ou executar planilhas.

Se considerarmos que um sistema operacional proprietário como o Windows XP Professional custa cerca de R$ 741,00 (setecentos e quarenta e um reais), e que uma suite de escritório como o Office Professional 2003 custa em torno de R$ 1.446,00 (mil quatrocentos e quarenta e seis reais), podemos então fazer uma conta simples: 741,00 + 1446,00 = 2.187,00 x 120.000 = R$ 262.440.000,00 (duzentos e sessenta e dois milhões quatrocentos e quarenta reais). Observe no entanto que este valor terá de ser novamente pago a medida que sai uma nova versão. Para entender melhor, imagine:

Você tem um amigo que mora nos Estados Unidos, e tanto você quanto seu amigo utilizam o mesmo editor de textos, chamado PALAVRA 98. Toda semana você monta um texto em um formato especial e envia para o seu amigo que mora em Boston, que ao receber seu texto por email, abre-o no editor PALAVRA 98, insere uma ou outra informação e publica o texto. Logo em seguida, ele envia de volta para você o material já com os textos acertados e o complemento que ele adicionou ao texto original. Você recebe o texto, imprime o material e guarda um backup para consulta futuras.

Um belo dia, o seu amigo americano, que tem mais condições financeiras que você, adquire uma nova versão do editor, chamada PALAVRA 2003 XisPa. Um negócio fabuloso de novo, mas que é basicamente o mesmo editor anterior só que com algumas implementações novas. Para lhe fazer uma surpresa, ele decide mexer no último texto que você enviou para ele e insere várias alterações e formatos diferentes, e decide salvar tudo isso no novo arquivo e enviá-lo novamente para você que está no Brasil com seu velho editor.

Ao chegar a noite, como você faz todos os dias, logo após a conexão da internet você baixa seus email e encontra o novo texto enviado por seu amigo. Ao tentar abrí-lo, o seu editor amigo reclama, dá erro, você vê uma página toda truncada, e por aí vai.

Este tipo de problema tão comum entre as versões dos softwares é uma das maneiras com que o mercado da informática obriga os seus clientes a comprarem novas versões dos seus produtos. Eles implementam alterações que só poderão ser acessadas, lidas, verificadas, a partir da aquisição de uma nova versão do produto. Isto significa que você terá de adquirir novamente uma nova versão, senão você não poderá continuar trabalhando com as outras pessoas que dividem com você as tarefas habituais na sua empresa ou na sua escola.

Observe que quando isso se trata apenas de uma pessoa, você terá um gasto a cada dois anos com uma nova versão do produto, no entanto, no nosso exemplo consideramos o Datasus com 120.000 micros, e com um gasto de R$ 262.440.000,00 (duzentos e sessenta e dois milhões

quatrocentos e quarenta mil reais) para cada vez que atualizar seu parque de softwares básicos. Isto significa que a cada dez anos o Datasus terá gasto cinco vezes o valor de compra, chegando ao final da década com gastos superiores a 2.6 Bilhões de reais apenas em atualizações de software.

Este tipo de gasto bi-anual é **anti-social**, pois faz com que o governo tenha que fazer investimentos caros em atualizações de programas de computador, quando poderia estar efetivamente gastando em programas sociais. Além disso, a forma com que a indústria do software trata seus usuários, obrigando-os a adquirir novas versões de seus programas é considerado **não ético**, pois além de ser um produto não escasso, o software é também considerado um bem intangível, ou seja, ele não é um bem que a medida que as pessoas o obtenham no mercado através de uma cópia ele se esgote em um determinado momento, como por exemplo uma mesa, uma cadeira ou um automóvel, que são compostos por materiais escassos, que num dado momento podem acabar.

Esta diferenciação faz com que o produto software seja um produto não rival em comparação a outras mercadorias, muito pelo contrario, nas palavras do Prof. Sérgio Amadeu, o software é um produto que quanto mais se distribui, quanto mais se é utilizado pelas pessoas, maior é o seu valor agregado.

2 - Divide o público e mantêm seus usuários desamparados e dependentes => É preciso observar um pouco a história do ponto de vista de um produto como os sistemas operacionais para que se possa entender o que ocorre no mercado de trabalho dentro da área da tecnologia da informação.

Inicialmente as empresas de informática como a IBM, UNISYS, MICROSOFT, dentre outras, possuíam equipamentos e sistemas operacionais que não eram conhecidos pela maioria dos técnicos, isto era um problema, pois o mercado profissional não tinham pessoas com conhecimento suficiente sobre o produto das empresas, o que causava uma tremenda dificuldade para a massificação dos computadores e dos sistemas operacionais. Empresas como a IBM e a UNISYS optaram por distribuir gratuitamente nas universidades os seus computadores e sistemas operacionais, criando assim um grande número de futuros profissionais que teriam conhecimento sobre os seus produtos, isto significaria que em pouco espaço de tempo, a medida que estes profissionais se formassem e fossem para o mercado de trabalho, levariam consigo as experiências adquiridas em computadores e sistemas operacionais que haviam praticado, fazendo com que, quando perguntados sobre "que computador você conhece?", ou "que sistema operacional você tem experiência" a resposta era sempre relativa ao conhecimento adquirido na universidade, fazendo com que estes profissionais se tornassem "garotos propaganda" de uma determinada empresa ou de um determinado produto chamado "sistema operacional".

Com a Microsoft a história era diferente, primeiro por que o mercado da empresa eram os computadores pessoais, e segundo, o mercado que a empresa desejava era muito maior. Isto fez com que fosse trilhado um outro caminho, muito parecido com as duas gigantes IBM e UNISYS, mas de uma forma diferente. Apesar de na época já existirem no mercado diversas alternativas para inibir a cópia dos produtos "softwares", a empresa permitiu que a cópia dos seus sistemas operacionais fossem feitas sem problemas, fazendo com que os usuários de PCs domésticos tivessem facilmente acesso ao seu produto, massificando-o rapidamente. Diferente de outras empresas como a APPLE, que ligavam o software diretamente ao hardware que era vendido apenas por eles no mercado, a Microsoft seguiu o caminho iniciado pela IBM, que desenhara um modelo de computador para ser acessível por todos, utilizando o conceito que hoje entendemos como Hardware Livre, ou seja, um modelo de computador que poderia ser industrializado por qualquer empresa de componentes eletrônicos. O que a Microsoft fez foi disponibilizar um sistema operacional que seria facilmente copiado para uso nestes computadores, não impedindo a cópia ou a "pirataria", como é chamada hoje.

Com o passar dos anos, e a massificação do seu produto, a empresa passou a ter o monopólio no mercado dos computadores pessoais, a grande maioria das empresas podia contratar pessoas no mercado de trabalho já com conhecimento do sistema operacional Windows, reduzindo assim os investimentos das empresas com a formação de pessoal. Vale a pena ressaltar que quando isso começou a ocorrer, a Microsoft criou os elementos que permitiria a ela cobrar incisivamente das empresas o custo dos seus produtos, utilizando para isso as legislações locais de cada país, tendo no Brasil a ABES (Associação Brasileira de Empresas de Software) o papel de descobrir, processar e cobrar judicialmente das empresas o devido valor de cada produto proprietário multiplicado por até 3000 (três mil) vezes, e até quatro anos de prisão para os responsáveis pela pirataria. Ou seja, o que serviu a Microsoft para fazer a propaganda do seu produto, hoje também serve para ser uma renda adicional à empresa, pela cobrança de multas absurdas.

Com tudo isso, podemos afirmar que hoje o mundo do software proprietário divide os usuários entre aqueles que tem poder aquisitivo para comprar um produto, e os demais que vivem a margem da lei, de forma irregular, como criminosos, que utilizam os produtos de forma ilegal.

Além deste aspecto, também existem os programas de computador que prendem os usuários por longos períodos, como é o caso de bancos de dados que possuem linguagens próprias, que são patenteadas e que não podem ser "disponibilizadas" para outros bancos de dados. Isto significa que se sua empresa optou por utilizar um banco de dados destes, e utilizou a linguagem proprietária, terá extrema dificuldade de migrar seus dados para outro banco, sem que seja necessário grandes investimentos para conversões. Por isso que afirmamos que o software proprietário divide o público e mantêm seus usuários dependentes.

2.2 O que é Software Livre ?

Ajudar outras pessoas é a base da nossa sociedade.

Desde o início da história, nos mais remotos tempos, o ser humano vem convivendo socialmente com os seus pares. Desde cedo o homem aprendeu que era muito mais fácil sobreviver junto com outros seres humanos do que sobreviver sozinho, além disso, ele também necessitava de uma outra característica, que era a adaptação. Com esses dois elementos, viver socialmente e adaptar-se, fez com que o ser humano sobrevivesse até os dias de hoje.

Um exemplo simples para entender como era necessário ao homem viver em conjunto: imagine que num dado momento da história de nossos ancestrais, o homem saía para caçar diariamente logo ao amanhecer, e num dado momento do dia ele retornava com um coelho ou um pequeno animal que era simples para um único homem capturar. Num dado momento, o homem percebeu que quando ele se juntava a outros homens e saíam para caçar juntos, eles retornavam à aldeia com um búfalo ou outro animal maior, permitindo assim que várias famílias se alimentassem sem problemas por vários dias, dando-lhes tempo livre para outras tarefas na aldeia.

Essa ação coletiva de compartilhar o alimento caçado por todos, ou de fazer algo junto a outros da mesma espécie, é tão natural para nós humanos quanto falar, respirar, etc. Faz parte do nosso dia a dia o convívio com outros da nossa espécie. Além do que, não existe um único ser humano em todo nosso planeta que não tenha sido ajudado ou ajudou alguém. É natural para todos nós vivermos juntos, produzirmos juntos, ser colaborativo, compartilhar, ajudar, dar uma força, enfim, ser solidário é parte do nosso cotidiano.

Wener

Assim foi o caminho da humanidade. Tudo que nossos ancestrais aprendiam era repassado para os filhos, dos filhos para os netos, e assim sucessivamente. O conhecimento sempre foi encarado por nós humanos como um instrumento de educação, um fato muito importante para a nossa sobrevivência. A musica, a escrita, a matemática e tantas outras informações foram criadas e disponibilizadas livremente para todos. Mas isso, até um determinado momento da nossa história foi assim, quando por volta do século 16, com o advento da máquina de imprimir produzida por GUTEMBERG, o Estado criou uma legislação para defender os inventores, os criadores, expressos nos termos da Propriedade Intelectual.

 Gutemberg

http://educaterra.terra.com.br/voltaire/cultura/gutemberg.htm

2.3 O que é Open Source ?

Em tradução literal significa "Fonte Aberta". De uma forma geral, as pessoas que se referem ao movimento do Software Livre como Open Source na verdade estão mais preocupadas em passar as qualidades mais técnicas, tais como: o fato de um programa ser aberto permite o estudo, os testes, a verificação das tarefas que ele executa, isso dá garantias melhores sobre as funcionalidades, garantindo a segurança, etc...

No geral, as pessoas que fazem parte deste movimento dito Open Source estão mais preocupadas com as questões técnicas, e não vêem no software livre uma alternativa de mudanças nas relações humanas.

Fazem parte deste grupo uma boa parte dos desenvolvedores do Kernel, o próprio Linus Torvalds, e um dos expoentes mais velhos, Eric S. Raymond, que escreveu "A Catedral e o Bazar".

2.4 O que é Free Software ?

Na tradução literal quer dizer "Programas de Computador Livres".

Quem se refere a "Software Livre" está trabalhando na perspectiva de dizer o mesmo que o mundo "Open Source" do ponto de vista técnica, mas também afirma que é necessário construirmos um mundo melhor, uma sociedade livre e solidária, uma nova sociedade baseada noutros princípios éticos, sendo, portanto, os entusiastas do movimento do software livre atores políticos, que vêem na tecnologia uma maneira de apoiarmos as mudanças no mundo para melhor.

A pessoa mais representativa deste grupo é o Richard Stallman, bem como a Distribuição Debian é a que melhor representa os preceitos do mundo livre, tendo uma referencia inquestionável na Filosofia GNU.

3 Reconhecendo liberdades...

3.1 As liberdades concedidas

Para que um software seja considerado livre, é necessário que a licença do software conceda ao usuário 4 liberdades básicas, a saber:

A liberdade de executar um programa, independente do propósito..

A liberdade de estudar um programa e adaptá-lo às suas necessidades.

A liberdade de distribuir cópias, tanto gratuitas quanto vendidas.

Aqui, nesta liberdade, vale a pena ressaltarmos que não existe uma dicotomia entre o software livre e o software pago. Os programadores de computador, assim como qualquer outro profissional, necessita sobreviver, pagar suas contas pessoais, o aluguel, a escola dos filhos, compras de supermercado, etc.

O fato de um software ser livre não deve ser confundido com software gratuito, vulgarmente conhecido como "freeware". Um software gratuito concede apenas a você o direito de utilizá-lo sem o pagamento de taxas ou licenças, já o software livre permite a você direitos muito além do valor das licenças.

Imagine que você vai até uma banca de jornais, compra uma revista por R$ 9,90 (nove reais e noventa centavos) que traz junto um encarte contendo um CD com uma distribuição de software livre, a Debian por exemplo. Ao chegar em casa, você faz cem cópias do CD, leva para a sua escola, e vende por R$ 5,00 (cinco reais) cada cópia. Isto é legal. Você também poderia distribuir gratuitamente para os seus amigos, para a empresa onde seus colegas trabalham, isso tudo seria muito legal, tanto do ponto de vista de uma ação solidária quanto do ponto de vista comercial.

Você também pode entrar numa loja de informática em um grande shopping, e lá comprar uma caixa produzida por uma das empresas que distribuem nacionalmente software livre, como a Conectiva por exemplo. Esta caixinha deve lhe custar algo em torno de R$ 200,00 (duzentos reais), mas em vez de ser apenas um CD com algumas páginas de revista falando do como instalar, a caixinha traz junto um manual de 300 páginas, um vaucher para atendimento de suporte por dois meses, um bottom de um pinguinzinho, uma camiseta e um adesivo para colocar no seu carro.

Note que em ambos os casos, o CD da revista e o CD que vem na caixinha da distribuição possuem software livre, a diferença está nos níveis de serviço ofertados ou nas coisinhas que acompanham a distribuição. Você tanto poderá copiar o CD da revista quanto o CD que vem junto a caixinha com os manuais e tudo mais, pois você estará, independente do dinheiro gasto, obtendo direitos que serão inalienáveis.

Ressalto ainda a importância da questão dos valores. Você sempre poderá adquirir um software livre de graça, ou atráves de uma revista, e até mesmo pagando um preço bem superior para ter acesso a um produto, mas seja lá o que você pague, se o produto for livre, você poderá fazer dele o que quiser, exceto uma única coisa: você não poderá, de forma alguma, transformar o que comprou em produto proprietário, transformá-lo em algo seu, exclusivamente. E por fim, a última liberdade:

A liberdade de distribuir versões modificadas, de tal forma que a comunidade se beneficie da sua produção intelectual.

Note que você pode comprar o CD do Debian numa banca de jornal, chegar em casa, escolher

um dos programas de computador que está contido na distribuição, alterá-lo do jeito que você achar interessante (adaptá-lo para as cores da sua empresa ou da sua escola, por exemplo), e redistribuir a versão modificada para os seus amigos e vizinhos, e até mesmo vender as suas alterações. Mas não esqueça: seja lá o que você faça, você nunca poderá transformar o programa em produto proprietário, em algo apenas seu, independente do tamanho das alterações que você tenha inserido.

3.2 Diferentes licenças

Existem basicamente dois tipos de licenças que tratam do resultado das criações intelectuais. O Copyright ("direito de cópia") e o Copyleft ("esquerdo de cópia").

O Copyright, como visto antes, é a licença que trata o mundo de forma proprietária, divide a sociedade, não é etico, impede a solidariedade, considera o conhecimento como um bem privado.

O Copyleft reconhece a autoria, permitindo a intervenção de terceiros, mantendo este direito à qualquer pessoa que venha a interferir no programa.

Basicamente, é como se você, após ter criado o programa, disponibiliza-o para o mundo dizendo: Concedo todos os direitos que tenho sobre este produto a terceiros, desde que estes direitos sejam mantidos aos demais.

Copyleft é uma concessão de direitos...

3.3 Licenças Livres

Existem várias licenças que são utlizadas no mundo do software livre, algumas 100% livres, como a **GPL** (General Public License) e a **FDL** (Free Documentation License), que são licenças publicadas pela Free Software Foundation.

Na página da FSF você encontrará uma lista com todas as licenças, definindo o que cada uma representa do ponto de vista da liberdade. **É imprescindível o conhecimento sobre cada uma delas**, evitando assim que você incorra no erro de utilizar um produto não livre.

4 Sistema Operacional e Kernel

4.1 O sistema GNU

GNU é o nome do sistema operacional criado por Richard Stallman no ano de 1984. Era muito comum na época os programadores darem nomes aos seus programas utilizando-se de acrônimos recursivos (acrônimo - sm., conjunto de letras, pronunciado como uma palavra normal, formado a partir das letras iniciais -ou de sílabas- de palavras sucessivas que constituem uma denominação), assim o Stallman a partir da afirmação "GNU Not Unix", criou o nome do sistema operacional, **GNU**.

4.2 O Kernel

No centro de um sistema operacional está o kernel. Ele é composto por vários programas que intermediam os aplicativos que você usa no seu sistema operacional e a máquina, o hardware.

Quando você manda o seu programa de edição de textos gravar um arquivo, ele faz uma chamada ao kernel e "diz para ele": grave estes dados!. O Kernel então verifica se existe espaço no disco, qual a primeira trilha livre para gravar, faz a gravação do seu arquivo, grava a tabela de índice do disco, retorna ao seu editor de textos e "diz para ele": gravei o arquivo!, ao que, para você, simplesmente a tela que você utilizou para mandar gravar o arquivo vai fechar (se tudo correu bem), se houver um erro na gravação, ou a falta de espaço em disco por exemplo, o seu aplicativo vai jogar outra tela no vídeo informando o "Erro na Gravação".

O Linux é um kernel, é um projeto iniciado em 25 de agosto de 1991 pelo finlandês Linus Torvalds, e foi apoiado por milhares de programadores ao redor do mundo.

O Hurd é um outro projeto, que vem sendo produzido pela Free Software Foundation, que utiliza um outro conceito, baseado em microkernel, implementando características diferentes.

O FreeBSD é um outro kernel, produzido pelo projeto BSD, e já existe projeto de implementá-lo junto a distribuição Debian.

Logo, GNU/Linux é o nome correto para o conjunto do Sistema Operacional GNU acrescido do kernel de Linux Torvalds. Se o Sistema Operacional GNU for acrescido do kernel FreeBSD, então o correto é chamar o conjunto de GNU/FreeBSD.

Note, o projeto de criação do Kernel de Linux Torvalds é de 1991, portanto, 7 anos após o projeto da Free Software Foundation ter iniciado. A GPL, o compilador GCC e o editor Emacs já estavam prontos e disponíveis juntamente com uma infinidade de programas que compunham o sistema operacional.

5 O Brasil e o Software Livre

5.1 Nossa realidade

Vivemos com um Estado sem recursos, temos dependência tecnológica (principalmente em sistemas operacionais) e a nossa população é carente de educação em tecnologia.

A proposta que vem sendo trabalhada por várias pessoas do movimento do software livre é a eliminação do analfabetismo tecnológico no Brasil, o Estado ter condições para incentivar e prover a sociedade com software livre, e incentivarmos a nossa sociedade a solidificar princípios como a **Solidariedade**, **Cooperação** e **Apoio Mútuo**.

5.2 Como fazer ?

Paralisação imediata da aquisição de software proprietário, pelo Estado (principalmente) e por entidades comprometidas com a sociedade.

Exigência de Software Livre, conforme as regras da GPL, tanto na produção interna das empresas, quanto nas aquisições de software no mercado.

APLICABILIDADE DOS RESULTADOS

Economia feita pelo Estado poderá ser investida em outros setores públicos como Educação, Segurança, Moradia, etc..

Brasil poderá retomar a produção de Software Nacional além dos meros aplicativos, tendo uma distribuição do Sistema Operacional com características próprias.

Empresas terão mais capital para investir em formação de pessoal, modernização dos parques computacionais, etc...

Os recursos gastos em Software permanecem no nosso país, reduzindo a evasão de divisas para o exterior.

5.3 Expectativa

O Software Livre é uma proposta de solução para os anseios sociais de uma tecnologia aberta, de uma sociedade justa e solidária, que viabilizará todos os segmentos sociais a ingressarem no terceiro milênio em condições de igualdade no conhecimento da informática.

6 Textos para leitura

6.1 O Projeto GNU - Richard Stallman

(Publicado originalmente no livro Open Sources)

Autor: Richard Stallman

Publicado pelo CIPSGA em: Junho de 2000

Por favor envie sua perguntas à FSF & GNU (em inglês) gnu@gnu.org.
Também há outros modos para contatar o FSF.

Por favor envie comentários sobre este artigo para webmasters@www.gnu.org, e envie outras perguntas para gnu@gnu.org. (em inglês)

Nota de Agradecimento

Quando o leitor se deparar com o texto produzido pelo Stallman, terá uma certeza imediata: Este não é um homem comum ! Richard Stallman é para muitos de nós que atuamos em movimentos sociais no Brasil, um revolucionário que tem o objetivo muito claro de transformar a sociedade a partir do uso da tecnologia.

Assim como ele, muitos outros homens incomuns encaram a luta cotidiana por mudanças no nosso país, que tem tantas desigualdades sociais. Elvino Bohn Gass (PT/RS) e um desses homens, que apesar de estar a frente da defesa da agricultura familiar e da reforma agrária no Estado do Rio Grande do Sul, teve a compreensao do quao importante e termos uma tecnologia solidaria e alternativa para a produçao do conhecimento brasileiro, e através do seu gabinete, encaminhou a tradução do texto do Stallman, que estamos publicando através do Comitê..

É nossa expectativa que açoes como esta do Deputado Elvino, ajudando a disseminar a filosofia do software livre durante o 1 Forum Inernacional (4 e 5 de maio de 2000 - Porto Alegre/RS), venham incentivar outras açoes nos demais gabinetes da esfera parlamentar estadual e federal do nosso país, em defesa da liberdade do conhecimento humano.

Djalma Valois Filho
Diretor Executivo CIPSGA
dvalois@cipsga.org.br

Indice

O Projeto GNU
por Richard Stallman

A primeira comunidade a compartilhar software
Quando eu comecei a trabalhar no Laboratório de Inteligência Artificial do MIT em 1971, tornei-me parte de uma comunidade que compartilhava software, já existente a vários anos. O ato de compartilhar software não estava limitado a nossa comunidade em particular; é tão antigo quanto os computadores, da mesma forma que compartilhar receitas é tão antigo como cozinhar. Mas nós fazíamos isto mais do que a maioria.

O Laboratório de IA usava um sistema operacional de timesharing denominado ITS (Incompatible Timesharing System [Sistema incompatível de tempo compartilhado]) que os hacers(1) da equipe do laboratório tinham projetado e escrito em linguagem Assembler para o PDP-10 da Digital, um dos maiores computadores da época. Como membro desta comunidade, um hacker de sistema na equipe do laboratório de IA, meu trabalho era para melhorar este sistema.

Nós não chamávamos nosso software de "software livre" porque este termo ainda não existia; mas isso é o que era. Quando alguém de outra universidade ou companhia queria portar e usar o programa, nós permitíamos isto com prazer. Se você visse alguém usando um programa interessante e pouco conhecido, sempre poderia pedir para ver o código-fonte, de forma que poderia lê-lo, mudá-lo, ou canibalizar certas partes do mesmo para fazer um novo programa.

O desmoronamento da comunidade
A situação mudou drasticamente durante o início dos anos 80 quando a Digital descontinuou a

série PDP-10. Sua arquitetura, elegante e poderosa nos anos 60, não se pôde estender naturalmente aos grandes espaços de endereçamento que ficaram possíveis nos anos 80. Isto significou que praticamente todos os programas que compuseram o ITS tornaram-se obsoletos.

A comunidade de hackers do laboratório de IA já tinha se desmoronado, algum tempo antes. Em 1981, a companhia subsidiária Symbolics tinha contratado quase todos os hackers do laboratório de IA, e a despovoada comunidade já não era mais capaz de se manter. (O livro "Hackers", de Steve Levy, descreve estes eventos, e mostra um panorama claro desta comunidade em seus primórdios). Quando o laboratório de IA adquiriu um novo PDP-10 em 1982, seus administradores decidiram utilizar o sistema de timesharing não livre da Digital em vez do ITS.

Os computadores modernos daquele tempo, como o VAX ou o 68020, tinham seus próprios sistemas operacionais, mas nenhum deles software livre: você tinha que assinar um "nondisclosure agreement" ("concordo em não revelar") até mesmo para obter uma cópia executável.

Isto significava que o primeiro passo para usar um computador era prometer que não ajudaria seu vizinho. Proibiu-se a existência de uma comunidade cooperativa. A regra feita pelos donos de software proprietário era: "se você compartilhar com seu vizinho, você é um pirata. Se quiser alguma mudança, peça-nos de forma que nós a façamos".

A idéia de que o sistema social do software proprietário --sistema que diz que você não tem permissão de compartilhar ou mudar o software-- é anti-social, que não é ético, que está simplesmente errado, pode ser uma surpresa para alguns leitores. Mas o que mais poderíamos dizer de um sistema que está baseado em dividir o público e manter os usuários desamparados ? Esses leitores que acham a idéia surpreendente podem ter levado o sistema social proprietário como determinado, ou julgam isto em função das condições sugeridas pelas companhias que fazem o software proprietário. "Software publishers" (editores de software) trabalharam muito tempo e duro para convencer as pessoas que há somente um modo de ver este tópico.

Quando os editores de software falam em "fazer valer" seus "direitos" ou em "parar a pirataria", isso que de fato dizem é secundário. A real mensagem destas declarações está nas suposições que eles dão por concedidas; é presumido que o público aceite sem crítica. Então vamos examiná-las.

Uma das suposições é que as companhias de software têm um direito natural inquestionável de serem donas do software e então terem poder sobre todos os seus usuários. (Se este fosse um direito natural, então não importa quanto dano causa ao público, nós não poderíamos contestar.) De modo muito interessante, a Constituição dos Estados Unidos de América e a tradição legal rejeitam esta visão; direito autoral não é um direito natural, mas um artifício que o governo impôs que limita o direito natural à cópia pelos usuários.

Outra suposição não declarada é que a única coisa importante sobre o software é qual tarefa ele permite você fazer --que nós usuários de computadores não deveríamos nos preocupar com que tipo de sociedade nos é permitido ter.

Uma terceira suposição é que nós não teríamos nenhum software utilizável (ou, nunca se teria um programa para fazer este ou aquele trabalho em particular) se nós não oferecêssemos a uma companhia poder sobre os usuários deste programa. Esta suposição pode ter soado plausível, antes do movimento para o software livre demonstrar que nós podemos fazer software bastante útil sem pô-los em correntes.

Se nos recusarmos a aceitar estas suposições, e julgarmos estes tópicos na base moral que nos dá o senso comum colocando o usuário em primeiro lugar, nós chegaremos a conclusões muito diferentes. Os usuários de computadores devem ter liberdade para modificar programas, para os

ajustar às suas necessidades, e liberdade para compartilhar software, porque ajudar outras pessoas é a base da sociedade.

Não há lugar aqui para nos estender no raciocínio que há atrás desta conclusão, e por esse motivo eu peço ao leitor que vá a página da web http://www.gnu.org/philosophy/why-free.es.html.

Uma escolha moral severa

Ao desaparecer minha comunidade, continuar como antes era impossível. Em vez disto, eu enfrentei uma escolha moral severa.

A escolha fácil era unir-me ao mundo do software proprietário, assinar os "acordos de não revelar" e prometer que não ajudaria meu companheiro hacker. Provavelmente eu também desenvolveria software que seria lançado debaixo de "acordos de não revelar" e desse modo também aumentado as pressões em outras pessoas de forma que eles traíssem seus companheiros.

Eu poderia ter feito dinheiro deste modo, e talvez me divertido escrevendo códigos. Mas eu sabia que ao término da minha carreira, ao olhar para atrás, anos construindo paredes para dividir as pessoas, sentiria que eu havia passado minha vida fazendo do mundo um lugar pior.

Já tinha experimentado ser o recebedor de um "acordo de não revelar", quando alguém recusou dar, a mim e ao Laboratório de IA do MIT, o código fonte para o controle de nossa impressora. (A ausência de certas características neste programa faziam com que o uso da impressora fosse extremamente frustrante.) Assim, eu não podia dizer a mim mesmo que os "acordos de não revelar" eram inocentes. Enfureceu-me muito quando ele recusou-se a compartilhar conosco; eu não pude dar a volta e fazer a mesma coisa a outra pessoa.

Outra escolha, direta mas desagradável, era abandonar o campo da informática. Desse modo minhas habilidades não seriam mal usadas, mas elas ainda seriam desperdiçadas. Eu não seria culpado de dividir e restringir os usuários de computadores, mas isto aconteceria igualmente.

Assim eu procurei o modo no qual um programador poderia fazer algo para o bem. Eu me perguntei: haveria algum programa ou programas que eu pudesse escrever, para tornar comunidade possível mais uma vez ?

A resposta era clara: a primeira coisa necessária era um sistema operacional. Este é o software crucial para começar a usar um computador. Com um sistema operacional você pode fazer muitas coisas; sem um, não consegue nem fazer funcionar o computador. Com um sistema operacional livre, nós poderíamos ter uma comunidade de hackers cooperando novamente --e convidar qualquer um para unir-se a nós. E qualquer um poderia usar um computador sem começar por conspirar para privar seus amigos.

Como um desenvolvedor de sistema operacional, eu tinha as habilidades apropriadas para esta tarefa. Assim, embora eu não tivesse garantias de sucesso, eu percebi que tinha sido escolhido para fazer esse trabalho. Eu decidi fazer com que o sistema fosse compatível com Unix porque deste modo seria portátil, e assim aqueles usuários de Unix poderiam migrar para ele com facilidade. O nome GNU foi escolhido seguindo uma tradição hacker, como acrônimo recursivo para "Gnu is Not Unix" (Gnu não é Unix).

Um sistema operacional não é só um kernel [núcleo], executando basicamente outros programas. Nos anos setenta, todo sistema operacional merecedor de ser chamado assim incluíam processadores de comandos, montadores, compiladores, interpretadores, depuradores, editores de

texto, programas de correio, e muitos mais. ITS os teve, Multics os teve, VMS os teve e Unix os teve. O Sistema Operacional GNU também os teria.

Mais tarde eu escutei estas palavras, atribuídas a Hillel(2): "Se eu não for por mim, quem será por mim?
Se eu só for por mim, o que eu sou?
Se não agora, quando?"

A decisão de começar o projeto GNU estava baseado em um espírito semelhante.

Free como liberdade"
O termo "free software" [em inglês free = livre ou grátis] às vezes é mal interpretado --não tem nada a ver com o preço. E sim com liberdade. Aqui, então, a definição de software livre é: um programa é software livre, para você, um usuário em particular, se:

Você tem liberdade para executar o programa, com qualquer propósito.
Você tem a liberdade para modificar o programa e adaptá-lo às suas necessidades. (Para fazer esta liberdade ser efetiva na prática, você deve ter acesso ao código fonte, porque modificar um programa sem ter a fonte de código é excessivamente difícil.)
Você tem liberdade para redistribuir cópias, tanto grátis como com taxa.
Você tem a liberdade para distribuir versões modificadas do programa, de tal modo que a comunidade possa beneficiar-se com as suas melhorias.

Como "free" [livre] refere-se a "freedom" [liberdade] e não a preço, não existe contradição entre a venda de cópias e o software livre. De fato, a liberdade para vender cópias é crucial: as coleções de software livre que são vendidos em CD-ROM são importantes para a comunidade e a venda, dos mesmos é um modo importante para obter fundos para o desenvolvimento de software livre. Então, se as pessoas não puderem incluir um programa nestas coleções, este não é um software livre.

Por causa da ambigüidade de "free", as pessoas a muito têm procurado alternativa, mas ninguém achou uma alternativa apropriada. O idioma inglês tem mais palavras e nuances que qualquer outro, mas falta uma palavra simples, não ambígua, palavra que signifique "free" [livre], como em "freedom" [liberdade] --"unfettered" [sem correntes] é a palavra que mais entra no íntimo significando. Outras alternativas como "liberated" [liberado], "freedom" [liberdade] e "open" [aberto] têm o significado errado ou alguma outra desvantagem.

O software GNU e o sistema GNU
O desenvolvimento de um sistema inteiro é um projeto muito grande. Para trazê-lo ao alcance, eu decidi adaptar e usar os pedaços existentes de softwares livres sempre que era possível. Por exemplo, eu decidi bem no inicio usar TeX como formatador de texto principal; poucos anos depois, decidi usar o Sistema X Window, em vez de escrever outro sistema de janelas para o GNU.

Por causa desta decisão, o sistema GNU não é o mesmo que a coleção de todos os softwares GNU. O sistema GNU inclui programas que não são nenhum software GNU, programas que foram desenvolvidos por outras pessoas e projetos para os seus próprios propósitos, o qual nós podemos usar porque eles são software livre.

Começando o projeto
Em janeiro de 1984 eu deixei meu trabalho no MIT e comecei a escrever o software GNU. Deixar o MIT era necessário para que o MIT não quisesse interferir com a distribuição de GNU como software livre. Se eu tivesse continuado como parte da equipe, o MIT, poderia ter reivindicado propriedade no trabalho, e poderia ter imposto as próprias condições de distribuição, ou até mesmo poderia transformar o trabalho em um pacote de software proprietário. Eu não tinha a intenção de fazer um trabalho enorme somente para vê-lo tornar-se inútil a seu almejado propósito: criar uma nova comunidade de software compartilhado.

Porém, o Professor Winston, então a cabeça do Laboratório de IA do MIT, gentilmente convidou-me a continuar usando a estrutura do laboratório.

Os primeiros passos
Pouco antes de começar no projeto GNU, eu escutei sobre o "Free University Compiler Kit" [Compilador da Universidade Livre], também conhecido como VUCK. (A palavra alemã para "free" começa com um V.) Era um compilador projetado para controlar múltiplas linguagens, inclusive C e Pascal, e para suportar máquinas de múltiplos propósitos. Eu escrevi ao autor perguntando se o GNU poderia usá-lo.

Ele me respondeu ironicamente, declarando que indubitavelmente a universidade era livre, mas o compilador não. Então, eu decidi que meu primeiro programa para o projeto GNU seria um compilador multilíngue e multiplataforma.

Com a esperança de evitar ter que escrever o compilador inteiro eu mesmo, obtive o código fonte do compilador Pastel, o qual era um compilador multiplataforma desenvolvido na "Lawrence Livermore Lab". Suportava e foi escrito em uma versão estendida de Pascal, projetado para ser usado como linguagem de programação em nível de sistemas. Eu acrescentei um "front end" em C, e comecei portando-o para o computador Motorola 68000. Mas eu tive que abandonar a idéia ao descobrir que o compilador precisava de muitos megabytes de espaço na pilha, e o sistema Unix para 68000 somente permitia 64 KB.

Eu percebi então que o compilador "Pastel" trabalhou analisando gramaticalmente o arquivo de entrada inteiro em uma árvore de sintaxe, convertendo essa árvore em uma cadeia de "instruções", e então gerando o arquivo de saída inteiro, sem liberar em qualquer momento o espaço ocupado. Neste momento, eu concluí que tinha que escrever um compilador novo partindo de zero. Esse novo compilador é agora conhecido como GCC; não há qualquer coisa do compilador "Pastel" nele, mas eu consegui adaptá-lo e usar o "front end" em C que tinha escrito. Mas isso aconteceu alguns anos depois; primeiro, eu trabalhei no GNU Emacs.

GNU EMACS
Eu comecei a trabalhar no GNU Emacs em setembro de 1984, e no começo de 1985 começou a ser usável. Isto me permitiu usar sistemas Unix para fazer a edição; não tendo nenhum interesse em aprender a usar o VI ou ED, eu tinha feito minha edição em outros tipos de máquinas até aquele momento.

A essas alturas, pessoas começaram a querer usar GNU Emacs o que levantou a pergunta de como distribuí-lo. Claro que, eu pus isto no servidor de FTP anônimo no computador do MIT que eu usava. (Este computador, prep.ai.mit.edu, transformado, se tornou assim o principal local de distribuição por FTP de GNU; quando foi confiscado depois de alguns anos, nós transferimos

o nome para nosso novo servidor de FTP.) Mas naquele tempo, muitas pessoas interessadas não estavam na Internet e não puderam obter uma cópia através de FTP. Assim a pergunta era: o que eu digo a eles?

Eu poderia ter dito, "ache um amigo que está na rede e que fará uma cópia para você". Ou poderiam ter feito o que eu fiz com o Emacs para PDP-10 original: lhes falai: "me envie uma fita e um envelope com o endereço e os selos de correio necessários, e eu lhe devolverei a fita com o Emacs dentro". Mas eu estava sem trabalho e estava procurando uma maneira de fazer dinheiro com o software livre. Então eu anunciei que enviaria uma fita para quem quisesse, por uma taxa de $150. Deste modo, eu comecei um negócio de distribuição de software livre, o precursor das companhias que atualmente distribuem sistemas completos GNU baseado em Linux.

O programa é livre para qualquer usuário?

Se um programa é software livre quando deixa as mãos de seu autor, isto não significa que será software livre para qualquer um que tenha uma cópia dele. Por exemplo, o software de domínio público (software que não está sujeito ao direito autorais de qualquer pessoa) é software livre; mas qualquer um pode fazer uma versão modificada proprietária dele. Igualmente, são registrados muitos programas livres mas distribuídos por meio de licenças simples que permitem versões modificadas proprietárias.

O exemplo paradigmatico deste problema é o sistema X Window. Desenvolvido no MIT, e liberado como software livre com um licença permissiva, foi logo adotado através de várias companhias de computador. Eles acrescentaram X a seus sistemas proprietários Unix, somente no formato binário, e coberto pelo mesmo "acordo de não revelar". Estas cópias de X não eram mais software livre do que o era o Unix.

Os desenvolvedores do sistema X Window não consideraram este um problema --eles esperavam e pretendiam que isto acontecesse. Sua meta não era liberdade, só o "sucesso", definido como "tendo muitos usuários". Não os preocupou se esses usuários teriam liberdade, só que eles deveriam ser numerosos.

Isto nos leva a uma situação paradoxal na qual dois modos diferentes de medir a liberdade deram respostas diferente à pergunta "é este um programa livre?". Se você julgasse baseado na liberdade provida pelas condições de distribuição do MIT, você diria que X é software livre. Mas se você medisse a liberdade do usuário comum de X, diria que X é software proprietário. A maioria dos usuários de X estava executando versões proprietárias que vieram dos sistemas Unix, não a versão livre.

Copyleft e o GNU GPL

A meta de GNU era dar liberdade aos usuários, não só ser popular. Então, nós deveríamos usar condições de distribuição que preveniriam que o software GNU se tornasse proprietário. O método que nós usamos foi denominado "copyleft"(3).

O copyleft usa a lei protegida por direitos autorais, mas dá a volta para servir ao oposto de seu propósito habitual: em vez de ser um meio de privatizar o software, se torna um meio de manter livre o software.

A idéia central do copyleft é que nós damos a qualquer um a permissão para executar o programa, copiar o programa, modificar o programa e redistribuir versões modificadas --mas nós não lhe damos permissão para somar restrições de sua propriedade. Deste modo, as liberdades

cruciais que definem o "software livre" são garantidos a qualquer um que tenha uma cópia; eles tornam-se direitos inalienáveis.

Para um copyleft efetivo, as versões modificadas também devem ser livres. Isto assegura que todo o trabalho baseado no nosso fica disponível para nossa comunidade se é publicado. Quando os desenvolvedores que trabalham como programadores voluntários para melhorar o software GNU, é o copyleft que impede que os empregadores digam: "não pode compartilhar essas mudanças, porque nós queremos usá-las para fazer nossa versão proprietária do programa".

A exigência de que as mudanças devem ser livres é essencial se nós quisermos assegurar a liberdade para cada usuário do programa. As companhias que privatizaram o sistema X Window em geral fizeram algumas mudanças para portar isto aos sistemas e ao hardware. Estas mudanças eram pequenas comparadas com o grande tamanho do X, mas elas não eram triviais. Se fazer mudanças fosse uma desculpa para negar liberdade aos usuários, seria fácil qualquer um tirar proveito da desculpa.

Um tópico relacionado trata a combinação de um programa livre com um de código não livre. Tal combinação será inevitavelmente não livre; qualquer liberdade que perdeu a parte não livre, também perderá o todo. Permitir tais combinações abriria um buraco grande o suficiente para afundar um navio. Para isto, é uma exigência crucial ao copyleft tapar este buraco: qualquer coisa somada ou combinada com um programa de copyleft deve ser de tal forma que a versão total combinada também seja livre e copyleft.

A implementação específica de copyleft que nós usamos para a maioria do software GNU é o "GNU General Public License" [GNU Licença de Público Geral] ou GNU GPL para abreviar. Nós temos outros tipos de copyleft que são usados em circunstâncias específicas. Manuais de GNU também são copyleft, mas usa um copyleft muito mais simples, porque a complexidade do GNU GPL não é necessário para manuais.

A Free Software Foundation (FSF)

Como o interesse no uso do Emacs foi crescendo, outras pessoas foram envolvidas no projeto GNU, e decidimos que estava na hora de procurar fundos novamente. Assim em 1985 criamos a "Free Software Foundation" [Fundação Software Livre], uma organização isenta de impostos para o desenvolvimento do software livre. A FSF também assumiu o negócio de distribuição em fita do Emacs; mais tarde estendeu isto ao acrescentar outros produtos de software livre (tanto GNU como não-GNU) para a fita, e com a venda de manuais livres também.

O FSF aceita doações, mas a maioria de suas rendas sempre veio das vendas --de cópias de software livre, e outros serviços relacionados. Hoje vende CD-ROMs de código fonte, CD-ROMs com binaries, manuais bem impressos (tudo com liberdade para redistribuir e modificar), e distribuições de luxo (onde nós incorporamos uma coleção inteira de software para a plataforma de sua escolha).

Os empregados da Fundação Software Livre escreveram e mantêm uma quantidade de pacotes de software GNU. Dois casos notáveis são a biblioteca de C e o Shell. A biblioteca GNU C é a usada por todo programa executado em um sistema GNU/Linux para comunicar-se com o Linux. Foi desenvolvido por um membro da equipe da Fundação, Roland McGrath. O Schell que é usado na maioria dos sistemas GNU/Linux é o bash, o Bourne Again Shell(4), que foi desenvolvido por Brian Fox, empregado do FSF.

Nós provemos os fundos para o desenvolvimento desses programas porque o projeto GNU não era só sobre ferramentas ou um ambiente de desenvolvimento. Nossa meta era um sistema

operacional completo, e esses programas eram necessários para essa meta.

Suporte ao Software Livre
A filosofia do software livre rejeita uma prática de negócio específica amplamente difundida, mas não está contra os negócios. Quando estes respeitam a liberdade dos usuários, nós lhes desejamos sucesso.

A venda de cópias de Emacs mostrou um tipo de negócio com software livre. Quando a FSF o assumiu, precisei de outro modo de ganhar a vida. Eu o encontrei na venda de serviços relacionada com o software livre que tinha desenvolvido. Isto incluiu ensino, assuntos de como programar GNU Emacs, e como personalizar GCC, e o desenvolvimento de software, na maioria portando, GCC para novas plataformas.

Hoje cada um desses tipos de negócio com software livre é praticado por várias corporações. Alguns distribuem coleções de software livre em CD-ROM; outros vendem suporte em níveis que variam desde respostas as questões do usuários, conserto de "bugs", até o agregado de novas características. Nós estamos até começando a ver companhias de software livre baseadas no lançamento de novos produtos de software livre.

Entretanto, tenha cuidado --várias companhias que se associam com o termo "Open Source" na realidade baseiam seus negócios em software não livre que trabalha com software livre. Elas não são companhias de software livre, mas companhias de software proprietário cujos produtos tentam os usuários para abandonar a liberdade. Usam a denominação "valor agregado" o que reflete os valores que eles gostariam que nós adotássemos: conveniência sobre liberdade. Se nós valorizássemos mais a liberdade, nós deveríamos denominar esses produtos de "liberdade subtraída".

Metas técnicas
A principal meta de GNU era ser software livre. Até mesmo se GNU não tivesse nenhuma vantagem técnica em cima do Unix, teria uma vantagem social, ao permitir cooperar com os usuários, e uma vantagem ética, respeitando a liberdade dos usuários.

Mas era natural aplicar os padrões conhecidos a boa prática do trabalho --por exemplo, alocando estruturas de dados dinamicamente para evitar limites arbitrários de tamanho fixo, e controlar todos os possíveis códigos de 8 bits onde quer que isso fizesse sentido.

Além disso, nós rejeitamos o foco do Unix em tamanhos de memória pequenos, decidindo por não dar suporte a máquinas de 16 bits (estava claro que as máquinas de 32 bits seriam a norma para quando o sistema GNU fosse acabado), e não fazer qualquer esforço para reduzir o uso de memória, a menos que excedesse o megabyte. Nos programas em que não era crucial a manipulação de arquivos muito grandes, nós incentivamos os programadores a ler o arquivo de entrada em memória, e então explorar o seu conteúdo, sem ter que preocupar-se com o E/S.

Estas decisões permitiram que muitos programas GNU ultrapassassem às compensações do UNIX em confiança e velocidade.

Computadores doados
Como a reputação do projeto GNU cresceu, as pessoas começaram a oferecer doações de

máquinas que executassem UNIX para o projeto. Estas eram muito úteis, porque o modo mais fácil de desenvolver componentes GNU era fazer isto em um sistema UNIX, e então substituir os componentes daquele sistema um a um. Mas eles trouxeram uma pergunta ética: se estava correto para nós ter uma cópia de todo a UNIX.

UNIX era (e é) um software proprietário, e a filosofia do projeto GNU diz que nós não deveríamos usar software proprietário. Mas, aplicando o mesmo raciocínio a estes objetivos concluímos que a violência em defesa é justificada, eu conclui que era legítimo usar um pacote proprietário quando isso era crucial para desenvolver uma substituição livre que ajudaria outros a deixar de usar o pacote proprietário.

Mas, mesmo se este fosse um mal justificável, ainda seria um mal. Hoje nós já não temos qualquer cópia de Unix, porque nós os temos substituído com sistemas operacionais livres. Se nós não pudéssemos substituir o sistema operacional de uma máquina por um livre, substituiríamos a máquina.

A lista de tarefas GNU

Como o projeto GNU prosseguiu, e foram achados um número crescente de componentes de sistemas ou foram desenvolvidos, eventualmente se tornou útil fazer uma lista das lacunas restantes. Nós usamos isto para recrutar desenvolvedores para escrever os pedaço que faltavam. Esta lista começou a ser conhecida como a lista de tarefas GNU. Além dos componentes Unix que faltavam, nós acrescentamos à lista vários outros softwares úteis e projetos de documentação que, nós pensávamos, deveria ter um sistema verdadeiramente completo.

Hoje, dificilmente algum componente Unix está na lista de tarefas GNU --esses trabalhos já foram acabados, fora alguns não essências. Mas a lista está cheia de projetos que alguns poderiam chamar "aplicações". Qualquer programa que atraia mais de uma classe estreita de usuários seria uma coisa útil para acrescentar a um sistema operacional.

Até mesmo jogos são incluídos na lista de tarefas --e estiveram desde o princípio. Unix incluiu jogos, assim naturalmente GNU também os incluiu. Mas a compatibilidade não era um assunto para os jogos, assim nós não seguimos a lista que teve o Unix. Ao invés disto, nós listamos uma gama de diferentes classes de jogos que os usuários pudessem gostar.

A Biblioteca GNU GPL

A biblioteca GNU C usa uma classe especial de copyleft denominado "GNU Library General Public License" [Licença Pública Geral para Bibliotecas GNU] isso dá permissão para conectar o software proprietário com a biblioteca. Porque fazer esta exceção?

Não é uma questão de princípios; nem há nenhum princípio que diga que produtos de software proprietário devam incluir nosso código. (Porque contribuir com um projeto que se recusa a compartilhar conosco?) O uso de LGPL para bibliotecas C, ou para qualquer outra biblioteca, é um assunto de estratégia.

A biblioteca C faz um trabalho genérico; todo o sistema proprietário ou compilador vem com uma biblioteca de C. Então, fazer nossa biblioteca só estar disponível para o software livre, não teria dado vantagem alguma --só teria desencorajado o uso da nossa biblioteca.

Há um sistema que é uma exceção a isto: no sistema GNU (e isto inclui os sistemas GNU/Linux), a biblioteca GNU C é a única biblioteca C. Assim as condições de distribuição da

biblioteca GNU C determinam se é possível compilar um programa proprietário para um sistema GNU. Não há nenhuma razão ética para permitir aplicações proprietárias no sistema GNU, mas estrategicamente parece que desaprovando, fará desencorajar mais o uso do sistema GNU que encorajar o desenvolvimento de aplicações livres.

Isso é por que o uso da biblioteca GPL é uma boa estratégia para a biblioteca C. Para outras bibliotecas, a decisão estratégica precisa ser considerada caso a caso. Quando uma biblioteca faz um trabalho especial que pode ajudar a escrever certos tipos de programas, então liberando-os abaixo do GPL, limitando-o só a programas livres, é um modo de ajudar a outros desenvolvedores de software livre, ao provê-los de uma vantagem contra o software proprietário.

Considere o GNU Readline, uma biblioteca desenvolvida para prover edição na linha de comando para bash. Readline é liberado debaixo do GNU GPL ordinário, não debaixo do GPL para Bibliotecas. Isto provavelmente diminui a quantidade de uso de Readline, mas isso não significa perda para nós. Enquanto isso, pelo menos uma aplicação útil foi feita especificamente para software livre assim pode-se usar Readline, e isso é um ganho real para nossa comunidade.

Os desenvolvedores de software proprietário têm as vantagens que o dinheiro provê; os desenvolvedores de software livre precisam criar vantagens um para o outro. Eu tenho a esperança de que algum dia nós tenhamos uma grande coleção de bibliotecas cobertas por GPL que não tenha paralelo disponível entre o software proprietário, provendo módulos úteis para servir como blocos construtivos em novos softwares livres, e acrescentando uma maior vantagem para adiantar o desenvolvimento de software livre.

"Quebrando um galho"?

Eric Raymond diz que "Todo o bom trabalho em software começa com um desenvolvedor quebrando um galho". Talvez isso aconteça às vezes, mas muitas das partes essenciais do software GNU foram desenvolvidas para ter um sistema operacional livre completo. Eles vieram de uma visão e um plano, não de um impulso.

Por exemplo, nós desenvolvemos a biblioteca GNU C porque um sistema do estilo Unix precisava de uma biblioteca C; o Bourne-Again Shell (bash) porque um sistema do estilo Unix precisava de um shell, e o GNU tar porque um sistema do estilo Unix precisava um tar. O mesmo é aplicado a meus próprios programas --o compilador GNU C, GNU Emacs, GDB e GNU Make.

Alguns programas GNU foram desenvolvidos para tratar ameaças específicas a nossa liberdade. Assim, nós desenvolvemos o gzip para substituir o programa de compressão, que estava perdido para nossa comunidade por causa das patentes da LZW. Nós achamos pessoas para desenvolver o LessTif, e mais recentemente começar o GNOME e o Harmony, para desviar os problemas causados por uma certa biblioteca proprietária (veja abaixo). Nós estamos desenvolvendo o GNU Privacy Guard [Guarda de Privacidade GNU] para substituir um popular software de criptografia não livre, porque usuários não devem ter que escolher entre privacidade e liberdade.

Claro que, as pessoas que escrevem estes programas tornaram-se interessadas no trabalho, e várias pessoas somaram muitas características a eles para satisfazer as suas próprias necessidades e interesses. Mas isso não é a razão para a qual os programas existem.

Desenvolvimentos inesperados

No começo do projeto GNU, imaginei que nós desenvolveríamos o sistema GNU inteiro, e então

liberá-lo como completo. Isso não foi o que aconteceu.

Considerando que cada componente de um sistema GNU foi implementado em um sistema Unix, todo componente poderia rodar em sistemas Unix, muito antes que existisse um sistema GNU completo. Alguns desses programas ficaram populares e os usuários começaram a os estender e os portar --para as várias versões incompatíveis de Unix, e às vezes para outros sistemas também.

O processo fez este programa muito mais poderoso, e atraiu fundos e contribuintes para o projeto GNU. Mas isso provavelmente também atrasou a conclusão de um sistema de funcionamento mínimo por muitos anos, como o tempo dos desenvolvedores GNU foi empregado em manter essa portabilidade e acrescentar características aos componentes existentes, em lugar de avançar em escrever um componente restantes.

O GNU Hurd

Em 1990, o sistema GNU estava quase completo; o único componente restante importante era o kernel. Nós decidimos implementar nosso kernel como uma coleção de processos servidores que rodam em Mach. Mach é um micro kernel desenvolvido na Carnegie Mellon University e depois na University of Utah; o GNU HURD é uma coleção de servidores (ou "rebanho de gnu´s") que rodam em Mach, e fazem as várias tarefas do kernel do Unix. O início do desenvolvimento foi atrasado enquanto nós esperávamos a liberação do Mach como software livre, como havia sido prometido.

Uma razão para isto era evitar o que parecia ser a parte mais dura do trabalho: depurar um kernel sem um depurador de código fonte para fazê-lo. Esta parte do trabalho já havia sido feita em Mach, e nós esperamos depurar os servidores HURD como programas de usuário, com GDB. Mas levou muito tempo para fazer isto possível, e os servidores multi-threaded que enviavam mensagens entre si mostraram-se muito difíceis de depurar. Fazendo o HURD trabalhar solidamente tinha esticado por muitos anos.

Alix

O kernel GNU não foi originalmente chamado HURD. O seu nome original era Alix --nomeado assim por causa da mulher que era minha amada naquele tempo. Ela, uma administradora de sistema Unix, havia mostrado como o seu nome se ajustaria aos padrão de nomenclatura comuns às versões de sistema Unix; por brincadeira, ela falou para seus amigos, "Alguém deveria dar o meu nome a um kernel". Eu não disse nada, mas decidi surpreendê-la com um kernel chamado Alix.

Não permaneceu desta maneira. Michael Bushnell (agora Thomas), o principal desenvolvedor do kernel, preferiu o nome HURD, e redefiniu Alix para se referir a uma certa parte do kernel --a parte que captura as chamadas do sistema e os negocia enviando mensagens para os servidores HURD.

No final da contas, Alix e eu nos separamos, e ela mudou seu nome; independentemente, o design de HURD foi mudado de forma que o biblioteca C enviaria mensagens diretamente aos servidores, e isto fez com que o componente Alix desaparecesse do design.

Mas antes de estas coisas aconteceram, um amigo dela deparou-se com o nome Alix no código fonte de HURD, e mencionou o nome a ela. Assim o nome cumpriu seu objetivo.

Linux e GNU/Linux
O GNU HURD não está pronto para o uso em produção. Felizmente, outro kernel está disponível. Em 1991, Linus Torvalds desenvolveu um kernel compatível com Unix e o chamou de Linux. Por volta de 1992, combinando Linux com o não completo sistema GNU, resultou num sistema operacional livre completo (claro que combiná-los foi um trabalho significativo). É devido ao Linux que atualmente nós podemos, de fato, rodar uma versão do sistema GNU .

Nós denominamos a esta versão GNU/Linux, para expressar a combinação do sistema GNU com Linux, como kernel.

Desafios em nosso futuro
Nós provamos nossa capacidade para desenvolver um largo espectro de software livre. Isto não significa que nós somos invencíveis e impossíveis de deter. Muitos desafios fazem o futuro do software livre incerto; enfrentá-los requererá esforços firmes e resistência, às vezes durante anos. Exigirá o tipo de determinação que as pessoas exibem quando valorizam sua liberdade e não permitirão que ninguém a tire.

As próximas quatro seções discutem estes desafios.

Hardware secreto
Os fabricantes de hardware crescentemente tentam manter segredo de suas especificações. Isto dificulta escrever drivers livres, para que Linux e XFree86 possam suportar assim, novos hardwares. Nós temos sistemas livres completos hoje, mas não os teremos amanhã se não pudermos suportar os computadores de amanhã.

Existem dois modos de lutar com este problema. Os programadores podem fazer engenharia reversa para entender como suportar o hardware. O resto de nós pode escolher o hardware que admite software livre; como nosso número aumenta, o segredo das especificações se tornará uma política derrotada.

A engenharia reversa é um grande trabalho; nós teremos programadores com suficiente determinação para empreender isto? Sim --se construirmos um sentimento forte de que o software livre é uma questão de princípio, e que os drivers não livres são intoleráveis. E um grande número de nós estará disposta a gastar dinheiro extra, ou até mesmo um pequeno tempo extra, para que possamos usar drivers livres? Sim, se a determinação de liberdade for difundida.

Bibliotecas não livres
Uma biblioteca não livre que roda em um sistema operacional livre atua como uma armadilha para os desenvolvedores de software livre. As características atraentes da biblioteca são a isca; se você usar a biblioteca, você cai na armadilha, porque seu programa não pode ser útil sendo parte de um sistema operacional livre. (No sentido exato, nós podemos incluir seu programa, mas não trabalhará sem o restante da biblioteca.) Pior ainda, se um programa que usa a biblioteca proprietária tornar-se popular, ela pode atrair outros programadores descuidados para a armadilha.

O primeiro exemplo deste problema era o equipamento de Motif Toolkit, lá nos anos oitenta.

Embora não houvesse ainda nenhum sistema operacional livre, estava claro o problema que Motif lhes causaria mais tarde. O projeto GNU respondeu de dois modos: solicitando a projetos individuais de software livre para suportar o Free X Toolkit Widgets tão bem quanto o Motif, e pedindo para alguém escrever uma substituição livre para o Motif. O trabalho levou vários anos; LessTif, desenvolvido pelos Hungry Programmers [os Programadores Famintos] ficou poderoso o bastante para suportar a maioria das aplicações Motif somente em 1997.

Entre 1996 e 1998, outra biblioteca de ferramentas GUI não livre, denominado Qt, era usado em uma coleção significativa de software livre: o desktop KDE.

Os sistemas livres GNU/Linux não puderam usar KDE, porque nós não podíamos usar a biblioteca. Porém, alguns distribuidores comerciais de sistemas GNU/Linux que não era tão rígido ao aderir ao software livre, acrescentaram o KDE aos seus sistemas --produzindo um sistema com mais capacidades, mas menos liberdade. O KDE Group estava encorajando ativamente a mais programadores a usar Qt, e milhões de novos "usuários Linux" nunca tinham sido expostos à idéia de que havia um problema nisto. A situação parecia severa.

A comunidade do software livre respondeu a este problema de dois modos: GNOME e Harmony.

GNOME, o GNU Network Object Model Environment [Ambiente de Modelagem de Objetos de Rede GNU], é o projeto GNU's desktop. Iniciado em 1997 por Miguel de Icaza, e desenvolvido com o suporte da Red Hat Software, GNOME teve a intenção de prover facilidades similares de desktop, mas usando exclusivamente software livre. Tem vantagens técnicas, tais como suportar uma variedade de linguagens, não só C++. Mas o seu principal propósito era a liberdade: não requerer o uso de qualquer software não livre.

Harmony é uma biblioteca de substituição compatível, projetada para tornar possível rodar o software KDE sem usar Qt.

Em novembro de 1998, os desenvolvedores de Qt anunciaram uma mudança de licença que quando levada a cabo, fará com que Qt seja software livre. Não há modo de estar seguro, mas eu penso que isto aconteceu em parte devido à resposta firme da comunidade frente ao problema que Qt apresentou quando não era livre. (A licença nova é inconveniente e injusta, assim permanece desejável evitar o uso de Qt.)

Como nós responderemos à próxima tentativa de biblioteca não livre? Irá toda a comunidade entender a necessidade de ficar fora da armadilha? Ou algum de nós desistirá da liberdade por conveniência, e gerará um problema maior? Nosso futuro depende de nossa filosofia.

Patente de software

A pior ameaça que nós enfrentamos são as patentes de software que podem colocar algoritmos e características fora dos limites do software livre por mais de vinte anos. A patente do algoritmo de compressão LZW foi pedido em 1983, e até agora o software livre não pode produzir GIFs comprimidos. Em 1998, um programa livre para produzir MP3 comprimido foi removido da distribuição sob a ameaça de uma termo de patente.

Há modos para lutar contra as patentes: nós podemos procurar evidência de que uma patente é inválida, e podemos procurar caminhos alternativos para fazer o trabalho. Mas cada um destes métodos só funciona algumas vezes; quando ambos falham, a patente pode forçar todo software livre a faltar com algumas características que os usuários querem. O que nós faremos quando isto acontecer?

Aqueles de nós que valorizam o software livre para a causa da liberdade ficará com o software livre de qualquer maneira. Nós nos prepararemos para ter nosso trabalho levado a cabo sem as características patenteadas. Mas esses que valorizam um software livre porque esperam que seja tecnicamente superior, é possível chamá-lo de falha quando uma patente o forçar a ficar atrás. Assim, embora seja útil falar sobre a efetividade prática do modelo "catedral" de desenvolvimento, e da confiança e poder de certo software livre, não deveríamos parar por aí. Temos que falar sobre liberdade e princípio.

Documentação livre

A maior deficiência em nosso sistema operacional livre não está no software --é a falta de bons manuais livres que nós possamos incluir em nossos sistemas. A documentação é uma parte essencial de qualquer pacote de software; quando um pacote importante de software livre não vem com um bom manual livre, fica uma grande lacuna. Nós temos atualmente muitas dessas lacunas.

A documentação livre, como o software, é uma questão de liberdade, não de preço. O critério para um manual livre é semelhante ao do software livre: é uma questão de conceder para os usuários certas liberdades. A redistribuição (até mesmo a venda comercial) deveria ser permitida, on-line e em papel, de tal modo que o manual possa acompanhar a toda cópia do programa.

A permissão para modificação também é crucial. Como regra geral, não acredito que é essencial que as pessoas tenham permissão para modificar todos os tipos de artigos e livros. Por exemplo, eu não penso que lhe ou me obriguem a dar permissão para modificar artigos como este que descreve nossas ações e nossa visão.

Mas uma razão particular existe por que a liberdade para modificar a documentação é crucial para o software livre. Quando as pessoas exercitam o seus direitos de modificar o software, e acrescentam ou mudam suas característica, se eles forem conscientes mudarão também o manual --eles proporcionarão deste modo a documentação precisa e útil ao programa modificado. Um manual que não permite os programadores serem conscienciosos e terminarem o trabalho, não preenche as necessidades de nossa comunidade.

A existência de alguns tipos de limites sobre como as modificações são feitas não possui problemas. Por exemplo, a exigência de preservar a advertência dos direitos autorais do autor original, os termos de distribuição, ou a lista de autores, estão O.K. Também não é nenhum problema requerer que a versão modificada inclua uma advertência de que foi modificado, e até mesmo ter seções inteiras que não podem ser apagadas ou modificadas contanto que estas seções tratem de tópicos não técnicos. Estes tipos de restrições não são um problema porque eles não impedem ao programador consciencioso de adaptar o manual para ajustar ao programa modificado. Em outras palavras, eles não impedem à comunidade do software livre o completo uso do manual.

Porém, deveria ser possível modificar todo o conteúdo técnico do manual, e então distribuir o resultado em todas as mídias usuais, por todos os canais habituais; caso contrário, as restrições obstruem a comunidade, o manual não é livre, e nós precisaremos de outro manual.

Irá o desenvolvedor de software livre ter a consciência e a determinação para produzir uma gama completa de manuais livres? Uma vez mais, nosso futuro depende de nossa filosofia.

Nós temos que falar sobre a liberdade
Estima-se hoje que haja aproximadamente dez milhões de usuários de sistemas GNU/Linux, como o Debian e Red Hat Linux. O software livre desenvolveu certas vantagens práticas que fazem os usuários estarem reunindo-se a ele por razões puramente práticas.

As conseqüências boas disto são evidentes: maior interesse no desenvolvimento de software livre, mais clientes para negócios de software livre, e mais habilidades para encorajar às companhias a desenvolver produtos de software livre, ao invés de produtos de software proprietário.

Mas o interesse pelo software cresce mais rápido que a consciência sobre a filosofia no qual é baseado, e isto conduz a problemas. Nossa capacidade para enfrentar os desafios e ameaças descritas acima depende da vontade de ficar firme pela liberdade. Para ter certeza de que nossa comunidade tenha essa vontade, nós precisamos difundir a idéia entre os usuários novos quando eles entram na comunidade.

Mas nós estamos falhando nisto: os esforços para atrair os usuários novos a nossa comunidade ultrapassam os esforços dedicados ao ensino cívico sobre a comunidade. Nós precisamos fazer ambos, manter os dois esforços equilibrados.

Open Source [Fonte Aberta]
O ensino sobre a liberdade para os usuários novos ficou mais difícil em 1998, quando uma parte da comunidade decidiu deixar de usar o termo "software livre" e usar "software de fonte aberto" no lugar dele.

Alguns favoreceram este termo, visando evitar a confusão de "livre" com "grátis" --uma meta válida. Porém, outros apontaram para fixar o espírito do princípio que motivou o movimento para o software livre e o projeto GNU, e ser deste modo atrativo aos executivos e usuários comerciais, muitos dos quais sustentam uma ideologia que coloca o lucro acima da liberdade, da comunidade, e dos princípios. Assim, a retórica de "fonte aberto" é focado no potencial de realização de software potente de alta qualidade, mas evita as idéias de liberdade, comunidade e princípio.

"As revistas de Linux" são um exemplo claro disto --elas estão cheias com anúncios sobre software proprietário que trabalha em GNU/Linux. Quando o próximo Motif ou Qt aparecer, estas revistas vão incentivar os programadores para ficar longe deles, ou colocarão propagandas do mesmo?

O apoio aos negócios pode contribuir à comunidade de vários modos; sendo tudo igual, isto é útil. Mas se ganhando o seu apoio mediante o recurso de falar menos sobre liberdade e princípio, pode ser desastroso; faz com que piore o desnível prévio entre o alcance e a educação cívica.

"Software livre" e "fonte aberto" descrevem a mesma categoria de software, mais ou menos, mas dizem coisas diferente do software, e sobre seus valores. O projeto GNU continua usando o termo "software livre" para expressar a idéia de que a liberdade, não só a tecnologia, é a coisa importante.

Tente!
A filosofia de Yoda ("there is no try" [não há tentativa]) soa bonito, mas não funciona comigo. Eu fiz a maioria de meu trabalho ansioso por não saber se conseguiria realizá-lo, e inseguro sobre se seria o bastante para alcançar a meta. Mas eu tentei igualmente, porque não havia outro entre o

inimigo e minha cidade. Para minha própria surpresa, às vezes tive sucesso.

Eu às vezes falhei; algumas de minhas cidades caíram. Então eu achei outro que ameaçou a cidade, e me preparei para outra batalha. Ao longo do tempo, aprendi como procurar as ameaças e me colocar entre eles e a minha cidade, chamando outros hackers a vir e unirem-se a mim.

Hoje em dia, freqüentemente eu não sou o único. É um alívio e um prazer quando eu vejo um regimento de hackers cavando trincheiras para manter a posição, e percebo que esta cidade pode sobreviver --por enquanto. Mas os perigos são maiores a cada ano, e agora a Microsoft tem a nossa comunidade como um alvo explícito. Nós não podemos ceder para garantir o futuro da liberdade. Não dê isso por certo! Se você quiser manter sua liberdade, deve estar preparado para defendê-la.

Referências:

(1) o uso de "hacker" para se referir ao "violador de segurança" é uma confusão que vem por parte dos meios de comunicação de massa. Nós hackers nos recusamos a reconhecer este significado, e continuamos usando a palavra para indicar "alguém que ama programar e que gosta de ser hábil e engenhoso".

(2) como ateísta, eu não sigo nenhum líder religioso, mas às vezes eu admiro alguma coisa que um deles disse.

(3) em 1984 ou 1985, Don Hopkins (um companheiro muito imaginativo) enviou-me uma carta. No envelope ele tinha escrito várias declarações divertidas, incluindo este aqui: "Copyleft-all rights reversed" [Copyleft--todo os direitos invertidos]. Eu usei a palavra "copyleft" para denominar o conceito de distribuição que estava desenvolvendo aquele tempo.

(4) "Bourne Again Shell" é uma brincadeira com o nome "Bourne Shell" que era o shell habitual em Unix.

Por favor envie sua perguntas à FSF & GNU (em inglês) gnu@gnu.org. Também há outros modos para contatar o FSF.
Por favor envie comentários sobre este artigo para webmasters@www.gnu.org, envie outras perguntas para gnu@gnu.org. (em inglês)

Artigo retirado da Internet em http://www.gnu.org
Atualizado em 20 /mar/2000

Tradução : Alexandre J. Thomé (Brasil) < alexandre.thome@intra.procergs.com.br >
Data: 12/abr/2000
Revisão Geral : Eliane M. de Azevedo (Brasil) < eliane.azevedo@intra.procergs.com.br >
Data: 13/abr/2000

6.2 The Right to Read - Richard Stallman

(Tradução de João Sebastião de Oliveiro Bueno) gwidion@email.com

Publicado no CIPSGA

Trabalhando na Área de design gráfico e finalizando a graduacao em Educação (Pedagogia), interessado em discutir a liberdade de expressão e da informação, dentro do contexto do debate entre a liberdade do individuo e dos interesses das grandes corporações, é uma das preocupações de José Sebastião de Oliveira Bueno, que escreveu a tradução deste texto.

The right to read" de RMS, na pagina da GNU - Tradução de: http://www.gnu.org/philosophy /right-to-read.html - Este artigo foi publicado na edição de fevereiro de 1997 de Communications of the ACM (Volume 40, Number 2 de "The Road to Tycho" , uma coleção de artigos sobre os antecedentes da Revolução Lunar, publicado em Luna City, em 2096)

Para Dan Halbert, o caminho para Tycho começou na faculdade, quando Lissa Lenz pediu seu computador emprestado. O dela havia quebrado,e , a não ser que ela conseguisse um outro emprestado, ela não conseguiria terminar seu projeto bimestral. E não havia ninguém a quem ela ousasse pedir isso, exceto Dan.

Isso deixou Dan num dilema. Ele tinha que ajuda-la, mas se emprestasse seu computador, ela poderia ler seus livros. Além do fato de que você pode ir para a prisão por muitos anos por deixar alguém ler seus livros, a própria idéia o chocou a princípio. Como todos mais, lhe tinham ensinado desde o primárioque emprestar livros era algo terrível e errado, algo que só piratas fariam.

E não havia muita chance de que a SPA - Software Protection Authority - não o descobrisse. Na sua aula de software, Dan havi aprendido que cada livro tinha embutido um monitor de copyright que informava quando e onde ele era lido, e por quem, para a Central de Licenciamento. (Eles usavam essa informação para pegar piratas de leitura, mas também para vender perfis de preferência de leitura para vendedores.) Na próxima vez em que seu computador estivesse conectado à rede, a Central de Licenciamento iria saber. Ele, como dono do computador, receberia a dura punição, por não ter feito os sacrifícios necessários para evitar o crime.

Claro que Lissa não pretendia, necessariamente, ler seus livros. Ele poderia quer o computador apenas para escrever seu projeto. Mas Dan sabia que ela vinha de uma família de classe média e mal podia arcar com as mensalidades, quanto mais suas taxas de leitura. Ler os livros de Dan poderia ser a única forma dela terminar o curso. (10% dessas taxas iam para os pesquisadores que escreviam os artigos; uma vez que Dan pensava em seguir carreira acadêmica, ele tinha esperanças de que seus próprios artigos de pesquisa, se fossem citados constantemente, renderiam o suficiente para pagar seu financiamento.)

Mais tarde, Dan aprenderia que havia um tempo em que qualquer pessoa poderia ir à biblioteca e ler artigos de periódicos, e até mesmo livros, sem ter que pagar. Haviam estudiosos independentes que liam milhares de páginas sem permissões governamentais para uso de biblioteca. Mas nos idos de 1990, editores tanto comerciais quanto institucionais de periódicos começaram a cobrar pelo acesso. Em 2047, bibliotecas oferecendo acesso gratuito ao público para artigos acadêmicos eram uma lembrança distante.

Havia formas, claro, de contornar a SPA e a Central de Licenciamento. Eram, eles mesmos, ilegais. Dan havia tido um colega na aula de software, Frank Martucci, que havia obtido uma ferramenta ilegal de depuração, a usava para pular o código monitor de copyright quando lia livros. Mas ele contou a muitos amigos sobre isso, e um deles o entregou à SPA por uma recompensa (estudantes devedores eram facilmente tentados pela traição). Em 2047, Frank estava na prisão, não por leitura pirata, mas por possuir um depurador.

Dan iria aprender depois que havia um tempo em qualquer pessoa podia ter ferramentas depuradoras. Havia até mesmo ferramentas depuradoras gratuitas disponíveis em CD, ou que podiam ser baixadas da
rede. Mas usuários normais começaram a usa-las para passar por cima dos monitores de copyright, e, eventualmente, um juíz declarou que isso havia se tornado seu uso principal na prática. Isso significava que elas se tornaram ilegais. Os desenvolvedores de ferramentas de depuração foram enviados para a prisão.

Programadores ainda precisavam de ferramentas de depuração, claro, mas vendedores de depuradores em 2047 distribuíam apenas cópias numeradas, e apenas para programadores oficialmente licenciados e juramentados. O depurador que Dan usou na aula de software era mantido atrás de uma firewall especial, de forma que podia ser usado somente para os exercícios da aula.

Também era possível passar por cima dos monitores de copyright instalando um kernl modificado do sistema operacional. Dan eventualmente descobriu sobre os kernels livres, e mesmo sistemas operacionais inteiros livres, que haviam existido por volta da virada do século. Mas eles não eram somente ilegais, como os depuradores, você não poderia instalar um mesmo que tivesse um, sem saber a senha do administrador do seu computador. E nem o FBI nem o Suporte da Microsoft lhe diriam qual ela é.

Dan concluiu que ele simplesmente não podia emprestar seu computador para Lissa. Mas ele não podia se recusar a ajuda-la, por que ele a amava. Cada chance de falar com ela o deixava em êxtase. E já que ela o havia escolhido para ajuda-la, isso poderia significar que ela o amava também.

Dan resolveu o dilema fazendo algo ainda mais impensável: ele emprestou seu computador a ela, e lhe disse sua senha. Dessa forma, se Lissa lesse seus livros, a Central de Licenciamento pensaria que ele os estava lendo. Isso ainda era um crime, mas a SPA não ficaria sabendo automaticamente sobre ele. Eles só saberiam se Lissa o entregasse.

Claro, se a faculdade descobrisse que ele tinha dado a Lissa sua própria senha, seria o fim para ambos enquanto estudantes, não importa para que ela tivesse usado essa senha. A política da faculdade era que qualquer interferência com as formas que ela tinha de monitorar o uso que os estudantes faziam do computador era o suficiente para ação disciplinar. Não importava se você havia feito qualquer coisa danosa, a ofensa tornava difícil que os administradores verificassem o que você estava fazendo. Eles assumiam que você estava fazendo alguma outra coisa que era proibida, e eles não precisavam saber o que era.
Alunos não eram expulsos por isso normalmente - não diretamente. Ao invés disso eles eram banidos do sistema de computadores da faculdade, e iriam, inevitavelmente, ser reprovados em seus cursos.

Depois, Dan aprenderia que esse tipo de política universitária havia começado apenas por volta dos anos 1980, quando os mais alunos começaram a usar os computadores. Anteriormente, as universidades tinham uma abordagem diferente para a disciplina; eles puniam atividades que eram danosas, não aquelas que meramente levantavam suspeitas.

Lissa não denunciou Dan para a SPA. Sua decisão de ajuda-la levou ao casamento dos dois, e também os levou a questionar o que eles tinham aprendido sobre pirataria enquanto crianças. O casal começou a aprender sobre a história do copyright, sobre a União Soviética e suas restrições para cópias, e mesmo sobre a Constituição original dos Estados Unidos. Eles se mudaram para Luna, onde eles encontraram outros que, da mesma forma, haviam gravitado para longe do longo braço da SPA. Quando o Levante de Tycho começou em 2062, o direito universal de leitura rapidamente se tornou um de seus objetivos centrais.

Nota do Autor
O direito de leitura é uma batalha que está sendo travada hoje. Embora ainda possa levar 50 anos para nossa forma corrente de vida desaparecer na obscuridade, a maior parte das leis e práticas descritas acima já foram propostas - Ou pela administração Clinton, ou por editores.

Há uma exceção: a idéia de que o FBI e a Microsoft terão a senha de administrador (root) dos computadores pessoais. Isso é uma extrapolação do Clipper chip e propostas similares da Administração Clinton, em conjunto com uma tendência a longo prazo: sistemas de computador estão mais e mais propensos a deixar o controle a operadores remotos do que a pessoas propriamente usando o sistema.

A SPA, que na verdade quer dizer 'Software Publisher's Association' (Associação dos Editores de Software), não é, hoje, uma força policial oficial. Extra-oficialmente, ela age como uma. Ela convida as pessoas a delatarem seus colegas de trabalho e amigos; como a Administração Clinton, ela advoga uma política de responsablidade coletiva na qual donos de computadores devem ativamente endossar os copyrights ou serem punidos.

A SPA está presentemente ameaçando pequenos provedores de serviço para a Internet, exigindo que eles permitam que a SPA monitore todos os usuários. A maioria dos provedores se rende quando ameaçada, por que eles não podem arcar com a batalha judicial. (Atlanta Journal-Constitution, 1 Oct 96, D3.) Pelo menos um provedor, Community ConneXion em Oakland CA, recusou a exigência e foi processado. Diz-se que a SPA desistiu desse processo recentemente, mas eles certamente continuarão sua campanha de várias outras formas.

As políticas de segurança de universidades descritas acima não são imaginárias. Por exemplo, um computador numa universidade na área de Chigago imprime esta mensagem quando você efetua o login (as aspas estão no original):

“Este sistema é para uso apenas de usuários autorizados. Indivíduos usando este sistema de computação sem permissão, ou excedendo sua permissão estão sujeitos a terem toda a sua atividade neste sistema monitorada e gravada pelo pessoal da administração. No caso de monitoramento de indivíduos fazendo uso incorreto desse sistema, ou no caso de manutenção do sistema, as atividades de usuários autorizados também poderá ser monitorada. Qualquer um usando o sistema expressamente consente com tal monitoramento, e é avisado de que tal se monitoramento revelar possíveis evidências de atividades ilegais, ou violação dos regulamentos da Universidade, a administração pode fornecer a evidência de tais atividades para autoridades da Universidade e/ou oficiais da lei. ¨

Esta é uma abordagem interessante para a Quarta Emenda [da constituição dos EUA]: pressiona quase todas as pessoas a concordarem, antecipadamente, a abdicar de seus direitos sob a mesma.

6.3 Linux e o Sistema GNU - Richard Stallman

Escrito por Richard Stallman
(sobre a relação entre o Linux e o projeto GNU)
Texto traduzido por Erik Kohler.

Publicado no CIPSGA

O projeto GNU começou há 12 anos atrás com o objetivo de desenvolver um sistema operacional Unix-like totalmente livre. "Livre" se refere à liberdade, e não ao preço; significa que você está livre para executar, distribuir, estudar, mudar e melhorar o software.

Um sistema Unix-like consiste de muitos programas diferentes. Nós achamos alguns componentes já disponíveis como softwares livres -- por exemplo, X Window e TeX. Obtemos outros componentes ajudando a convencer seus desenvolvedores a tornarem eles livres -- por exemplo, o Berkeley network utilities. Outros componentes nós escrevemos especificamente para o GNU -- por exemplo, GNU Emacs, o compilador GNU C, o GNU C library, Bash e Ghostscript. Os componentes desta última categoria são "software GNU". O sistema GNU consiste de todas as três categorias reunidas.

O projeto GNU não é somente desenvolvimento e distribuição de alguns softwares livres úteis. O coração do projeto GNU é uma idéia: que software deve ser livre, e que a liberdade do usuário vale a pena ser defendida. Se as pessoas têm liberdade mas não a apreciam conscientemente, não irão mantê-la por muito tempo. Se queremos que a liberdade dure, precisamos chamar a atenção das pessoas para a liberdade que elas têm em programas livres.

O método do projeto GNU é que programas livres e a idéia da liberdade dos usuários ajudam-se mutuamente. Nós desenvolvemos software GNU, e conforme as pessoas encontrem programas GNU ou o sistema GNU e comecem a usá-los, elas também pensam sobre a filosofia GNU. O software mostra que a idéia funciona na prática. Algumas destas pessoas acabam concordando com a idéia, e então escrevem mais programas livres. Então, o software carrega a idéia, dissemina a idéia e cresce da idéia.

Em 1992, nós encontramos ou criamos todos os componentes principais do sistema exceto o kernel, que nós estávamos escrevendo. (Este kernel consiste do microkernel Mach mais o GNU HURD. Atualmente ele está funcionando, mas não está preparado para os usuários. Uma versão alfa deverá estar pronta em breve.)

Então o kernel do Linux tornou-se disponível. Linux é um kernel livre escrito por Linus Torvalds compatível com o Unix. Ele não foi escrito para o projeto GNU, mas o Linux e o quase completo sistema GNU fizeram uma combinação útil. Esta combinação disponibilizou todos os principais componentes de um sistema operacional compatível com o Unix, e, com algum trabalho, as pessoas o tornaram um sistema funcional. Foi um sistema GNU variante, baseado no kernel do Linux.

Ironicamente, a popularidade destes sistemas desmerece nosso método de comunicar a idéia GNU para as pessoas que usam GNU. Estes sistemas são praticamentes iguais ao sistema GNU -- a principal diferença é a escolha do kernel. Porém as pessoas normalmente os chamam de "sistemas Linux (Linux systems)". A primeira impressão que se tem é a de que um "sistema Linux" soa como algo completamente diferente de "sistema GNU", e é isto que a maioria dos

usuários pensam que acontece.

A maioria das introduções para o "sistema Linux" reconhece o papel desempenhado pelos componentes de software GNU. Mas elas não dizem que o sistema como um todo é uma variante do sistema GNU que o projeto GNU vem compondo por uma década. Elas não dizem que o objetivo de um sistema Unix-like livre como este veio do projeto GNU. Daí a maioria dos usuários não saber estas coisas.

Como os seres humanos tendem a corrigir as suas primeiras impressões menos do que as informações subsequentes tentam dizer-lhes, estes usuários que depois aprendem sobre a relação entre estes sistemas e o projeto GNU ainda geralmente o subestima.

Isto faz com que muitos usuários se identifiquem como uma comunidade separada de "usuários de Linux", distinta da comunidade de usuários GNU. Eles usam todos os softwares GNU; de fato, eles usam quase todo o sistema GNU; mas eles não pensam neles como usuários GNU, e frequentemente não pensam que a filosofia GNU está relacionada a eles.

Isto leva a outros problemas também -- mesmo dificultando cooperação com a manutenção de programas. Normalmente quando usuários mudam um programa GNU para fazer ele funcionar melhor em um sistema específico, eles mandam a mudança para o mantenedor do programa; então eles trabalham com o mantenedor explicando a mudança, perguntando por ela, e às vezes reescrevendo-a para manter a coerência e mantenebilidade do pacote, para ter o patch instalado.

Mas as pessoas que pensam nelas como "usuários Linux" tendem a lançar uma versão "Linux-only" do programa GNU, e consideram o trabalho terminado. Nós queremos cada e todos os programas GNU que funcionem "out of the box" em sistemas baseados em Linux; mas se os usuários não ajudarem, este objetivo se torna muito mais difícil de atingir.

Como deve o projeto GNU lidar com este problema? O que nós devemos fazer agora para disseminar a idéia de que a liberdade para os usuários de computador é importante?

Nós devemos continuar a falar sobre a liberdade de compartilhar e modificar software -- e ensinar outros usuários o valor destas liberdades. Se nós nos beneficiamos por ter um sistema operacional livre, faz sentido para nós pensar em preservar estas liberdades por um longo tempo. Se nós nos beneficiamos por ter uma variedade de software livres, faz sentido pensar sobre encorajar outras pessoas a escrever mais software livre, em vez de software proprietário.

Nós não devemos aceitar a idéia de duas comunidades separadas para GNU e Linux. Ao contrário, devemos disseminar o entendimento de que "sistemas Linux" são variantes do sistema GNU, e que os usuários destes sistemas são tanto usuários GNU como usuários Linux (usuários do kernel do Linux). Usuários que têm conhecimento disto irão naturalmente dar uma olhada na filosofia GNU que fez estes sistemas existirem.

Eu escrevi este artigo como um meio de fazer isto. Outra maneira é usar os termos "sistema GNU baseado em Linux (Linux-based GNU system)" ou "sistema GNU/Linux (GNU/Linux system)", em vez de "sistema Linux", quando você escreve sobre ou menciona este sistema.

6.4 Richard Stallman e Bradley M. Kuhn : " Liberdade ou Poder ? "

Publicado no CIPSGA

"O amor à liberdade é o amor ao próximo; o amor ao poder é o amor a nós mesmos."
William Hazlitt

No movimento pelo Software Livre, nós lutamos por liberdade para os usuários de software. Nós formulamos nossas visões ao examinar quais liberdades são necessárias para um bom modo de vida, e para permitir a programas úteis fomentar uma comunidade de boa-vontade, cooperação e colaboração. Nossos critérios para Software Livre especificam as liberdades que o usuário de u programa necessita de modo que ele possa cooperar em uma comunidade.

Nós lutamos por liberdade para os programadores assim como para os demais usuários. A maioria de nós são programadores, e nós queremos liberdade para nós também. Mas cada um de nós utiliza software escrito por outros, e nós queremos liberdade quando utilizamos estes softwares, não apenas quando utilizamos o nosso próprio código. Nós lutamos por liberdade para todos os usuários, sejam eles programadores frequentes, ocasionais, ou não sejam programadores em absoluto.

Entretanto, uma tão-falada liberdade que nós não defendemos é a "liberdade de escolher qualquer licensa que você deseje para o software que você escreve". Nós rejeitamos isto porque é na verdade uma forma de poder, não de liberdade.

Esta distinção, frequentemente negligenciada, é crucial. Liberdade é ser capaz de tomar decisões que afetam principalmente a você mesmo. Poder é ser capaz de tomar decisões que afetam a outros mais do que a você. Se nós confundirmos poder com liberdade, nós falharemos em sustentar a verdadeira liberdade.

Software proprietário é um exercício de poder. A lei de Copyright atual garante ao desenvolvedor de software este poder, de modo que ele e somente ele pode escolher as regras impostas sobre todos os outros -- relativamente poucas pessoas tomando as decisões básica sobre o software para todos, tipicamente negando as suas liberdades. Quando os usuários não tem as liberdades que definem o Software Livre, elas não podem dizer o que o software está fazendo, não podem verificar se existem back doors, não podem monitorar possíveis vírus e vermes, não podem descobrir se informações pessoais estão sendo enviadas para alguém (ou parar os envios, se elas os descobriem). Se ele parar de funcionar, elas não podem consertar; elas tem que esperar que o desenvolvedor execise o seu poder de fazer o conserto. Se ele simplesmente não é o que os usuários necessitavam, eles tem que se contentar com isso. Elas não podem se ajudar uns aos outros a aperfeiçoar o software.

Os desenvolvedores de software proprietário são em geral empresas. Nós no Movimento pelo Software Livre não somos contra as empresas, mas nós já vimos o que acontece quando uma empresa de software tem a "liberdade" de impor regras arbitrárias sobre os usuários do software. A Microsoft é um exemplo notório de como negar as liberdades aos usuários pode causar danos diretos, mas ela não é o único exemplo. Mesmo quando não há monopólio, o software proprietário prejudica a sociedade. A escolha de mestres não é liberdade.

Discussões sobre direitos e regras para o software foram frequentemente concentradas apenas nos direitos dos programadores. Poucas pessoas no mundo programam regularmente, e ainda menos são proprietárias de empresas de software. Mas todo o mundo desenvolvido hoje necessita e utiliza software, de modo que os desenvolvedores de software hoje controlam como o

mundo vive, faz negócios, se comunica e se diverte. As questões políticas e éticas não são atendidas pelo slogan "liberdade de escolha (somente para os desenvolvedores)".

Se o código é lei, como o Professor Lawrence Lessig (da Escola de Direito de Stanford) afirma, então a verdadeira questão que nós enfrentamos é: quem deveria controlar o código que você utiliza -- você, ou uma pequena elite? Nós acreditamos que você tem o direito de controlar o software que você utiliza, e dar a você este controle é o objetivo do Software Livre.

Nós acreditamos que você deveria decidir o que fazer com o software que você utiliza; entretanto, isto não é o que as leis atuais dizem. As leis atuais de Copyright nos colocam em posição de poder sobre os usuários do nosso código, gostemos ou não disto. A resposta ética a esta situação é proclamar a liberdade para cada usuário, assim como a Lei dos Direitos foi criada para que o governo exercesse o poder pela garantia da liberdade de todos os cidadãos. É para isto que serve a GNU GPL: ela coloca você no controle da utilização do software, enquanto que protege você de outros que gostariam de tomar o controle sobre as suas decisões.

À medida que mais usuários compreenderem que código é lei, e descobrirem que eles também desejam liberdade, eles irão ver a importância das liberdades pelas quais nós lutamos, assim como mais e mais usuários aprenderam a apreciar o valor prático do Software Livre que nós desenvolvemos.

Fonte:http://www.fsf.org