O projeto GNU

The GNU project *

Richard Stallman**

Resumo: Quando comecei a trabalhar no Laboratório de Inteligência Artificial do MIT, em 1971, incorporei-me a uma comunidade que já compartilhava programas há muitos anos. O ato de compartilhar software não se limitava à nossa comunidade em particular, é algo tão velho como o computador, do mesmo modo que compartilhar receitas é tão antigo como cozinhar. Mas nós o fazíamos em uma escala maior do que a maioria. (...) A idéia de que o sistema social do software proprietário -- o sistema que diz que você não pode compartilhar ou trocar programas -- é anti-social, de que não é ético, de que simplesmente é algo errado, pode surpreender alguns leitores. Mas, que outra coisa poderíamos dizer de um sistema que se baseia na divisão do público e que mantém os usuários sem socorro? Os leitores que se surpreendem com esta idéia devem ter assimilado o sistema social do software proprietário tal como lhes foi dado, ou o julgaram em função dos termos sugeridos pelo negócio do software proprietário. Os que publicam software têm trabalhado longa e duramente para convencer as pessoas de que só há um modo de se considerar este tema. Quando os que publicam software falam de "fazer valer" os seus "direitos" ou de "deter a pirataria", o que *dizem*, de fato, é secundário. A verdadeira mensagem contida nessas declarações está nos pressupostos não declarados que eles consideram garantidos; o público deve aceitá-los acriticamente. (...) Os usuários de computador devem ser livres para modificar programas, de forma a ajustá-los às suas necessidades, e livres para compartilhar software, porque a base da sociedade está em ajudar às outras pessoas. (...) Quando a minha comunidade desapareceu, foi impossível continuar como antes. (...) Perguntei-me, então, se havia algum programa ou programas que pudesse escrever, de modo a tornar outra vez possível uma

comunidade. A resposta era clara: o que se necessitava, em primeiro lugar, era um sistema operacional. Este é o software crucial para se poder começar a usar um computador. Com um sistema operacional pode-se fazer muitas coisas; sem ele, você não pode nem fazer funcionar o computador. Com um sistema operacional livre, poderíamos ter novamente uma comunidade cooperativa de *hackers* [1] -- e convidar qualquer pessoa para juntar-se a ela. E qualquer um seria capaz de usar um computador, sem ter para isso que conspirar contra os seus amigos e amigas. (...) Decidi fazer o sistema compatível com o Unix, de modo que fosse portátil e, assim, que os usuários do Unix pudessem adotá-lo facilmente. O nome GNU foi escolhido segundo uma tradição *hacker*, como um acrônimo recursivo de «*GNU's Not Unix*». (...) Não podemos considerar o futuro da liberdade garantido. Não o considere garantido! Se você deseja conservar a sua liberdade, deve estar preparado para defendê-la.

Palavras Chave: GNU, Software Livre, Patentes de Software, Software Proprietário, Liberdade de Informação, Copyright

Abstract: When I started working at the MIT Artificial Intelligence Lab in 1971, I became part of a software-sharing community that had existed for many years. Sharing of software was not limited to our particular community; it is as old as computers, just as sharing of recipies is as old as cooking. But we did it more than most.(...) The idea that the proprietary software social system--the system that says you are not allowed to share or change software--is antisocial, that it is unethical, that it is simply wrong, may come as a surprise to some readers. But what else could we say about a system based on dividing the public and keeping users helpless? Readers who find the idea surprising may have taken proprietary social system as given, or judged it on the terms suggested by proprietary software businesses. Software publishers have worked long and hard to convince people that there is only one way to look at the issue. When software publishers talk about "enforcing" their "rights" or "stopping piracy", what they actually *say* is secondary. The real message of these statements is in the unstated assumptions they take for granted; the public is supposed to accept them uncritically. (...)Computer users should be free to modify programs to fit their needs, and free to share software, because helping other people is the basis of society. With my community gone, to continue as before was impossible. (...) I asked myself, was there a program or programs that I could write, so as to make a

community possible once again? The answer was clear: what was needed first was an operating system. That is the crucial software for starting to use a computer. With an operating system, you can do many things; without one, you cannot run the computer at all. With a free operating system, we could again have a community of cooperating hackers [1] --and invite anyone to join. And anyone would be able to use a computer without starting out by conspiring to deprive his or her friends. (...) I chose to make the system compatible with Unix so that it would be portable, and so that Unix users could easily switch to it. The name GNU was chosen following a hacker tradition, as a recursive acronym for "GNU's Not Unix." (...) We can't take the future of freedom for granted. Don't take it for granted! If you want to keep your freedom, you must be prepared to defend it.

Keywords: GNU, Free Software, Software Patents, Proprietary Software, Freedom of Information,

A primeira comunidade que compartilhava programas

Quando comecei a trabalhar no Laboratório de Inteligência Artificial do MIT, em 1971, incorporei-me a uma comunidade que já compartilhava programas há muitos anos. O ato de compartilhar software não se limitava à nossa comunidade em particular, é algo tão velho como o computador, do mesmo modo que compartilhar receitas é tão antigo como cozinhar. Mas nós o fazíamos em uma escala maior do que a maioria.

O Laboratório de IA usava um sistema operacional de tempo compartilhado, denominado ITS (*Incompatible Timesharing System* - sistema incompatível de tempo compartilhado), que os *hackers* [1] da equipe haviam projetado e escrito em linguagem de montador para o PDP-10 da Digital, um dos maiores computadores existentes na época. Como membro daquela comunidade, como um *hacker* da equipe de sistema do laboratório de IA, o meu trabalho era melhorar aquele sistema.

Não chamávamos os nossos programas de "software livre", uma vez que tal expressão ainda não existia; mas era isso o que eles eram. Quando alguém de outra universidade ou companhia desejava levar e usar um programa, nós o permitíamos alegremente. Ao ver alguém usando um programa interessante e pouco familiar para você, era sempre possível pedir para ver o código fonte, lê-lo, modificá-lo ou canibalizar parte dele para fazer um

novo programa.

[1] O uso da palavra "*hacker*" para descrever alguém que "quebra a segurança" é uma confusão proveniente da mídia. Nós, os *hackers*, nos negamos a reconhecer tal acepção e continuamos a usar a palavra com o sentido de "alguém que tem paixão por programar e que adora ser engenhoso ao fazê-lo".

O colapso da comunidade

A situação mudou drasticamente nos primeiros anos 80, quando a Digital descontinuou a série PDP-10. A sua arquitetura, elegante e poderosa na década de 60, não pôde ser estendida, naturalmente, aos espaços de endereçamento mais amplos que se tornaram factíveis nos anos 80. Isto significava que quase todos os programas que compunham o ITS estavam obsoletos.

A comunidade *hacker* do laboratório de IA já havia colapsado, não fazia muito tempo. Em 1981, a companhia emergente Symbolics havia contratado a quase todos os *hackers* do laboratório de IA, e a comunidade despovoada já não era capaz de se manter. (O livro *Hackers*, de Steve Levy, descreve aqueles acontecimentos e dá uma idéia clara da comunidade em seus primórdios.) Quando o laboratório de IA adquiriu um novo PDP-10, em 1982, os seus administradores resolveram usar o sistema de tempo compartilhado proprietário [*non-free*] da Digital, ao invés do ITS.

Os computadores modernos daquela época, como o VAX ou o 68020, têm os seus próprios sistemas operacionais, mas nenhum deles era software livre: você tinha que assinar um acordo de não divulgação [*nondisclosure agreement*], mesmo que fosse para obter uma cópia executável.

Isto significava que o primeiro passo para se poder utilizar um computador era prometer não ajudar a seu vizinho. Uma comunidade cooperativa estava proibida. A regra estabelecida pelos donos do software proprietário era a de que "se você compartilha com o seu vizinho, você é um pirata. Se desejar qualquer mudança, peça-nos para fazê-la". A idéia de que o sistema social do software proprietário -- o sistema que diz que você não pode compartilhar ou trocar programas -- é anti-social, de que não é ético, de que simplesmente é algo errado, pode surpreender alguns leitores. Mas, que outra coisa poderíamos dizer de um sistema que se baseia na divisão do público e que mantém os usuários sem socorro? Os leitores que se surpreendem com esta idéia devem ter assimilado

o sistema social do software proprietário tal como lhes foi dado, ou o julgaram em função dos termos sugeridos pelo negócio do software proprietário. Os que publicam software têm trabalhado longa e duramente para convencer as pessoas de que só há um modo de se considerar este tema. Quando os que publicam software falam de "fazer valer" os seus "direitos" ou de "deter a pirataria", o que *dizem*, de fato, é secundário. A verdadeira mensagem contida nessas declarações está nos pressupostos não declarados que eles consideram garantidos; o público deve aceitá-los acriticamente. Vamos examiná-los, então.

Um pressuposto é que as companhias de software têm um direito natural inquestionável à propriedade dos programas e, assim, de dispor de poder sobre todos os seus usuários. (Nada poderíamos objetar, independentemente do dano que causasse ao público, se isso fosse um direito natural.) O interessante é que a Constituição dos EUA e a tradição legal rejeitam este ponto de vista; o copyright não é um direito natural, mas um monopólio artificial imposto pelo governo que limita o direito natural de copiar do usuário.

Outro pressuposto não declarado é que a única coisa importante no software é o trabalho que ele permite a você realizar -- que a nós, usuários de computadores, não nos deve importar que tipo de sociedade nos é permitido ter.

Um terceiro pressuposto é que não teríamos qualquer software utilizável (ou que não existiria jamais um programa para esta ou aquela tarefa) se não oferecêssemos a uma empresa poder sobre os usuários do programa. Este pressuposto pode ter parecido plausível antes de que o movimento pelo software livre demonstrasse que podemos fazer uma quantidade imensa de programas úteis sem impor cadeias a eles.

Se nos recusamos a aceitar tais pressupostos e consideramos esses temas sob o fundamento moral que o senso comum ordinário nos oferece, colocando o usuário em primeiro lugar, chegaremos a conclusões muito diferentes. Os usuários de computadores devem ter a liberdade de modificar programas para adaptá-los às suas necessidades e a liberdade de compartilhar software, uma vez que a base da sociedade consiste em ajudar-nos uns aos outros.

Não dispomos aqui do espaço necessário para um tratamento extenso das razões por trás desta conclusão, de modo que remeto o leitor para a página web http://www.gnu.org/philosophy/why-free.html.

Uma severa escolha moral

Uma vez desaparecida a minha comunidade, continuar como antes já não era possível. Em vez disso, enfrentei um severo dilema moral.

A escolha mais fácil era aderir ao mundo do software proprietário, assinar os acordos de não divulgação e prometer não ajudar o meu companheiro *hacker*. Provavelmente eu também desenvolveria software a ser distribuído sob acordos de não divulgação, fazendo aumentar desse modo a pressão sobre outros para que também traíssem os seus companheiros.

Poderia ter ganhado dinheiro deste modo e, talvez, me divertido escrevendo código. Mas eu sabia que, no fim da minha carreira, olharia para os anos dedicados a construir paredes para dividir as pessoas e sentiria que havia gastado a minha vida tornando o mundo um lugar pior.

Eu já tinha experimentado o outro lado de um acordo de não divulgação, quando alguém se havia recusado a entregar, a mim e ao Laboratório de IA do MIT, o código fonte do programa de controle da nossa impressora. (A falta de certas características naquele programa fazia com que o uso da impressora fosse extremamente frustrante.) Logo, eu não podia dizer-me a mim mesmo que os acordos de não divulgação eram inocentes. Fiquei muito aborrecido quando ele se recusou a compartilhar conosco; eu não podia dar meia volta e fazer o mesmo com as outras pessoas.

Outra escolha, direta mas desagradável, era abandonar o campo da computação. Assim, as minhas habilidades não seriam mal empregadas, embora, de qualquer modo, fossem desperdiçadas. Eu não seria culpado por dividir e restringir usuários de computador, mas isto aconteceria, apesar de tudo.

Procurei então um modo pelo qual um programador pudesse fazer algo pelo bem.Perguntei-me, então, se havia algum programa ou programas que pudesse escrever, de modo a tornar outra vez possível uma comunidade.

A resposta era clara: o que se necessitava, em primeiro lugar, era um sistema operacional. Este é o software crucial para se poder começar a usar um computador. Com um sistema operacional pode-se fazer muitas coisas; sem ele, você não pode nem fazer funcionar o computador. Com um sistema operacional livre, poderíamos ter novamente uma comunidade cooperativa de *hackers* -- e convidar qualquer pessoa para juntar-se a ela. E

qualquer um seria capaz de usar um computador, sem ter para isso que conspirar contra os seus amigos e amigas.
Como desenvolvedor de sistemas operacionais, eu possuia as habilidades adequadas para a tarefa. De modo que, embora sem ter absoluta certeza do êxito, percebi que havia sido eleito para fazer aquele trabalho. Decidi fazer o sistema compatível com o Unix, de modo que fosse portátil e, assim, que os usuários do Unix pudessem adotá-lo facilmente. O nome GNU foi escolhido segundo uma tradição *hacker*, como um acrônimo recursivo de «*GNU's Not Unix*».
Um sistema operacional não significa somente um núcleo, apenas suficiente para fazer rodar outros programas. Nos anos 70, todo sistema operacional digno desse nome incluía processadores de comando, montadores, compiladores, interpretadores, depuradores, editores de texto, programas de correio e muito mais. O ITS tinha, o Multics tinha, o VMS tinha e o Unix tinha. O sistema operacional GNU também iria incluí-los.
Mais tarde, ouvi estas palavras, atribuídas a Hillel [2]:

"Se não preocupo comigo mesmo, quem vai fazê-lo por mim?
Se me preocupo apenas comigo mesmo, o que sou?
Se não for agora, quando?"

A decisão de iniciar o projeto GNU se baseou em um espírito similar.

[2] Como ateu que sou, não sigo nenhum líder religioso, mas às vezes acho que admiro alguma coisa dita por algum deles.

"Free" como "livre" [N.T.: em inglês, *free* tanto significa *livre* como *grátis*]
A expressão "*free software*" é às vezes mal compreendida -- ela não tem nada que ver com preço. Trata-se aqui de liberdade. Eis, portanto, a definição de software livre: um programa é um "*free software*" para você, um usuário particular, se

- Você tiver a liberdade de executar o programa, com qualquer propósito.

- Você tiver a liberdade de modificar o programa para adaptá-lo às suas necessidades. (Para tornar tal liberdade efetiva na prática, você deve ter acesso ao código fonte, uma vez que modificar um programa sem ter o código fonte é excessivamente difícil.)
- Você tiver a liberdade de redistribuir cópias, seja gratuitamente, seja por uma taxa.
- Você tiver a liberdade de distribuir versões modificadas do programa, de tal modo que a comunidade possa ser beneficiada com os seus aperfeiçoamentos.

Como "free" se refere a liberdade e não a preço, não há contradição entre a venda de cópias e o software livre. De fato, a liberdade de vender cópias é crucial: as coleções de software livre que se vendem em CD-ROMs são importantes para a comunidade, e a sua venda é um modo importante de se levantarem fundos para o desenvolvimento de software livre. Portanto, um programa que não possa ser incluído livremente nessas coleções não é software livre.

Devido à ambiguidade do termo "free", buscaram-se outros, mas ninguém encontrou até agora uma alternativa apropriada. O inglês tem mais palavras e nuances do que qualquer outra língua, mas lhe falta uma palavra simples e não ambígua que signifique "livre", como em liberdade -- sendo o termo "*unfettered*" ["sem grilhões"] o que mais se aproxima daquele sentido. Outras alternativas, como "*liberated*" ["liberado"], "*freedom*" ["liberdade"] e "*open*" ["aberto"] têm um significado equivocado ou alguma outra desvantagem.

Software GNU e o sistema GNU

O desenvolvimento de um sistema completo é um projeto bem grande. Para torná-lo factível, decidi adaptar e usar partes já existentes de software livre sempre que possível. Por exemplo, desde o princípio, decidi usar o TeX como o principal formatador de texto; poucos anos mais tarde, decidi usar o sistema X Window, ao invés de escrever outro sistema de janelas para o GNU.

Por causa disso, o sistema GNU não coindide com o conjunto de todo o software GNU. O sistema GNU inclui programas que não são software GNU, programas que foram desenvolvidos por outras pessoas e projetos para os seus próprios propósitos, mas que podemos usar por serem software livre.

Começando o projeto

Em janeiro de 1984, deixei o meu emprego no MIT e comecei a escrever o software GNU. Abandonar o MIT era necessário, para que o MIT não pudesse interferir com a distribuição do GNU como software livre. Se eu tivesse permanecido na equipe, o MIT poderia ter reivindicado a propriedade do trabalho e ter imposto as suas próprias cláusulas de distribuição, ou ainda tê-lo transformado em um pacote de software proprietário. Eu não tinha nenhuma intenção de realizar um trabalho imenso e vê-lo depois tornar-se inútil para os propósitos originais: criar uma nova comunidade que compartilhasse software. Contudo, o professor Winston, responsável então pelo Laboratório de IA do MIT, convidou-me, gentilmente, a continuar usando as instalações do laboratório.

Os primeiros passos

Um pouco depois de começar o projeto GNU, ouvi falar do *Free University Compiler Kit* [Kit Compilador da Universidade Livre], também conhecido como VUCK (em holandês, *free* se escreve com V). Era um compilador projetado para tratar com várias linguagens, inclusive C e Pascal, e para dar suporte a várias máquinas de destino. Escrevi ao seu autor, perguntando se o GNU poderia usá-lo.

Ele me respondeu zombeteiramente, dizendo que a universidade era *free*, mas o compilador não era. Decidi, por isso, que o meu primeiro programa para o projeto GNU seria um compilador multi-linguagem e multi-plataforma.

Com a esperança de evitar o trabalho de escrever todo o compilador por conta própria, consegui o código fonte do Pastel, um compilador multi-plataforma desenvolvido no *Lawrence Livermore Lab*. Ele dava suporte e estava escrito em uma versão estendida de Pascal, projetada para ser uma linguagem de programação de sistema. Adicionei um *front end* para C e comecei a transportá-lo para o computador Motorola 68000. Mas tive que desistir, quando descobri que o compilador necessitava muitos megabytes de espaço de pilha, enquanto que o sistema Unix 68000 disponível permitia somente 64k.

Foi quando percebi que o compilador Pastel funcionava analisando o arquivo de entrada inteiro em uma árvore sintática, convertia a árvore sintática completa em uma cadeia de "instruções" e, então, gerava o arquivo de saída inteiro, sem nunca liberar qualquer espaço de armazenamento.

Naquele momento, cheguei à conclusão de que deveria escrever um novo compilador a

partir do zero. Esse novo compilador é conhecido agora como GCC; não há nada nele do compilador Pastel, mas dei um jeito de adaptar e usar o *front end* para C que já havia escrito. Mas isto só aconteceu uns anos depois; primeiro, trabalhei no GNU Emacs.

GNU Emacs

Comecei a trabalhar no GNU Emacs em setembro de 1984, e já no começo de 1985 ele começava a ser utilizável. Isto me permitiu começar a usar os sistemas Unix para edição; até aquele momento, eu havia feito meus trabalhos de edição em outros tipos de máquina, uma vez que me faltava qualquer interesse em aprender a usar o *vi* ou o *ed*. Naquela altura, as pessoas começaram a querer usar o GNU Emacs, o que levantou a questão de como distribuí-lo. Certamente, o coloquei no servidor de FTP anônimo do computador do MIT que eu usava. (Esse computador, prep.ai.mit.edu, tornou-se assim o site ftp principal de distribuição do GNU; quando foi desativado uns anos mais tarde, transferimos o nome para o nosso novo servidor ftp.) Mas, naquele tempo, muitas das pessoas interessadas não estavam na Internet e não poderiam obter uma cópia por ftp. A questão, então, era o que dizer a eles. Poderia haver dito, "procure um amigo que esteja na rede e que faça uma cópia para você". Ou poderia ter feito o que fiz com o Emacs para PDP-10 original, poderia dizer-lhes: "envie-me uma fita e um envelope com o seu endereço, que eu o retornarei com o Emacs dentro". Mas eu não estava trabalhando e procurava formas de fazer dinheiro com software livre. Assim, anunciei que poderia enviar uma fita pelo correio para qualquer pessoa que assim o desejasse, por uma taxa de $150. Desse modo, comecei um negócio de distribuição de software livre, o precursor das companhias que hoje distribuem sistemas completos GNU baseados em Linux.

Um programa é livre para qualquer usuário?

Se um programa é software livre quando sai das mãos do seu autor, isto não significa necessariamente que será software livre para todos os que possuam uma cópia do programa. Por exemplo, o software de domínio público (software sem registro de copyright) é software livre; mas qualquer um pode fazer uma versão modificada proprietária do mesmo. De modo semelhante, muitos programas livres que estão registrados sob copyright são

distribuídos mediante licenças simples permissivas, que admitem versões modificadas proprietárias.

O exemplo paradigmático desse problema é o sistema *X Window*. Desenvolvido no MIT e distribuído como software livre com uma licença permissiva, foi logo adotado por várias companhias de computação. Elas acrescentaram o *X* aos seus sistemas Unix proprietários, somente em forma binária e coberto pelo mesmo acordo de não divulgação. Aquelas cópias do *X* não eram mais software livre do que o Unix.

Os desenvolvedores do sistema *X Window* não viam nenhum problema nisso -- é o que esperavam e procuravam que acontecesse. O seu objetivo não era a liberdade, apenas o "êxito", definido como "tendo muitos usuários". Não lhes importava se esses usuários tinham liberdade, apenas que fossem numerosos.

Isto leva a uma situação paradoxal, na qual duas formas diferentes de avaliar a quantidade de liberdade produzem diferentes respostas à pergunta "este programa é livre?". Se julgar com base na liberdade oferecida pelas cláusulas de distribuição do MIT, você diria que *X* era software livre. Mas, se medir a liberdade do usuário médio do *X*, você teria que dizer que era software proprietário. A maioria dos usuários de *X* esteve usando as versões proprietárias que vêm com os sistemas Unix, e não a versão livre.

Copyleft e a GNU GPL

O objetivo do GNU era dar liberdade aos usuários, não apenas ser popular. Portanto, precisávamos usar cláusulas de distribuição que impedissem o software GNU de se tornar software proprietário. O método utilizado se denomina "*copyleft*" [3] .

O *copyleft* usa a lei de *copyright* [*right* = direita, direito], mas a inverte para servir a um propósito oposto ao usual: em vez de instrumento para privatizar o software, torna-se um meio de mantê-lo livre.

A idéia central do *copyleft* é dar a qualquer pessoa autorização para executar o programa, copiá-lo, modificá-lo e distribuir cópias modificadas -- mas sem permitir que adicione restrições próprias. Desse modo, as liberdades cruciais que definem o "software livre" permanecem garantidas para todos aqueles que possuam uma cópia; tornam-se direitos inalienáveis.

Para um *copyleft* efetivo, as versões modificadas também precisam ser livres. Isto garante

que todo trabalho baseado no nosso se torna disponível para a nossa comunidade, uma vez publicado. Quando programadores que têm empregos como programadores se oferecem como voluntários para aperfeiçoar um software GNU, é o copyleft que impede os seus empregadores de dizer: "você não pode compartilhar estas modificações, pois vamos usá-las para fazer a nossa versão proprietária do programa".
O requisito de que aquelas mudanças devem ser livres é essencial, se quisermos garantir a liberdade de todo usuário do programa. As companhias que privatizaram o sistema *X Window* fizeram geralmente algumas alterações para transportá-lo aos seus sistemas e hardware. Tais modificações foram pequenas, se comparadas com o grande tamanho do *X*, mas de forma alguma triviais. Se o fato de fazer alterações fosse uma desculpa para recusar liberdade aos usuários, seria fácil para qualquer um tirar vantagem da desculpa.
Um problema similar diz respeito à combinação de um programa livre com um código não livre. Uma tal combinação será, inevitavelmente, não livre; qualquer liberdade que falte à parte não livre, faltará também ao todo. Permitir tais combinações abriria um buraco de dimensões suficientes para afundar um navio. Por isso, um requisito crucial para o *copyleft* é que este buraco esteja tampado: qualquer coisa que venha a ser agregada ou combinada com um programa sob *copyleft* deve ter características tais que a versão composta maior esteja igualmente livre e sob *copyleft*.
A implementação específica de *copyleft* que usamos para a maior parte do software GNU é a *GNU General Public License* [Licença Pública Geral GNU] ou, abreviadamente *GNU GPL*. Temos outros tipos de *copyleft* que são usados em circunstâncias específicas. Os manuais GNU também estão sob *copyleft*, mas um *copyleft* muito mais simples, já que as complexidades do *GNU GPL* não são necessárias para manuais.

[3] Em 1984 ou 1985, Don Hopkins (uma pessoa muito imaginativa) enviou-me uma carta. No envelope havia várias expressões divertidas, entre as quais a seguinte: "*Copyleft -- all rights reversed*" [todos os direitos "invertidos"]. Usei a palavra "*copyleft*" para denominar o conceito de distribuição que eu estava desenvolvendo na época. [N.T.: *left* = esquerda; deixado (do verbo *leave*, deixar; *with your leave* = com a sua licença)]

A Fundação para o Software Livre

À medida que o interesse no uso do Emacs aumentava, outras pessoas se envolveram com o projeto GNU e decidimos que já era hora de procurar fundos novamente. Por isso, em 1985 criamos a "*Free Software Foundation*" (*FSF*) [Fundação para o Software Livre], uma organização filantrópica, isenta de impostos, para o desenvolvimento do software livre. A *FSF* também se encarregou do negócio da distribuição de fitas com o Emacs; o que ampliou mais tarde pela inclusão de outros programas livres (tanto GNU como não GNU) nas fitas, assim também como pela venda de manuais livres.

A FSF aceita doações, mas a maior parte de seus ingressos tem vindo sempre das vendas -- de cópias de software livre e de outros serviços correlatos. Hoje, vende CD-ROMs de códigos-fonte, CD-ROMs com binários, manuais primorosamente impressos (todos com liberdade de redistribuição e de modificação) e Distribuições De Luxo (nas quais incluimos todo o conjunto de software para a plataforma de sua escolha).

Os funcionários da *Free Software Foundation* escrevem e mantêm certo número de pacotes de software GNU. Dois deles, notáveis, são a biblioteca C e o shell. A biblioteca C do GNU é o que todo programa que roda em um sistema GNU/Linux usa para se comunicar com o Linux. Ela foi desenvolvida por um membro da equipe da *Free Software Foundation*, Roland McGrath. O shell usado na maior parte dos sistemas GNU/Linux é o BASH, o *Bourne Again SHell*[4], desenvolvido por Brian Fox, funcionário da *FSF*.

Financiamos o desenvolvimento desses programas porque o projeto GNU não se tratava apenas de ferramentas ou de ambiente de desenvolvimento. Nossa meta era um sistema operacional completo, e esses programas eram necessários para aquele objetivo.

[4] "*Bourne Again SHell*" é uma brincadeira com o nome "*Bourne Shell*", que era o shell usual do Unix. [N.T.: born again = nascer de novo; Bourne é o autor do shell usual do Unix]

Suporte para o Software Livre

A filosofia do software livre rejeita certa prática específica de negócios amplamente difundida, mas não se posiciona contra os negócios. Quando os negócios respeitam a liberdade dos usuários, desejamos que tenham êxito.

A venda de cópias do Emacs exemplifica um tipo de comércio de software livre. Quando a

FSF assumiu aquele negócio, eu precisei de um outro meio de vida. Consegui isso vendendo serviços relacionados com o software livre que havida desenvolvido. O que incluia o ensino, tal como programação do GNU Emacs e como personalizar o GCC, e o desenvolvimento de software, em sua maior parte destinado a transportar o GCC para novas plataformas. Hoje, cada um destes tipos de comércio de software livre é praticado por certo número de corporações. Algumas distribuem coleções de software livre em CD-ROM; outras vendem serviços de apoio, em níveis que vão desde responder a perguntas de usuários até o conserto de defeitos [*bugs*] e o acréscimo de novas características importantes. Estamos inclusive começando a ver companhias de software livre com base no lançamento de novos produtos de software livre.

Mas tenha cuidado -- um certo número de companhias que se associam, elas mesmas, com a expressão "*open source*" ["código-fonte aberto"] baseiam, de fato, os 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 atraem os usuários para fora do âmbito da liberdade. Elas os denominam "valor agregado", o que reflete os valores que gostariam que adotássemos: conveniência acima da liberdade. Se damos à liberdade maior valor, deveríamos chamá-los de produtos com "liberdade subtraída".

Objetivos técnicos

O objetivo principal do GNU era ser software livre. Mesmo que o GNU não apresentasse vantagens técnicas relativamente ao Unix, ele teria uma vantagem social, ao permitir que os usuários cooperassem entre si, e uma vantagem ética, ao respeitar a liberdade dos usuários. Mas era natural aplicar ao trabalho os padrões conhecidos da boa prática -- por exemplo, alocar dinamicamente as estruturas de dados para que se evitassem limites arbitrários de tamanho fixo, e manipular todos os códigos possíveis de 8 bits sempre que isso fizesse sentido.

Além disso, rejeitamos o foco em pequeno tamanho de memória adotado pelo Unix, ao decidirmos não tratar com máquinas de 16 bits (estava claro que as máquinas de 32 bits seriam a norma no momento em que o sistema GNU estivesse pronto) nem fazer nenhum esforço para reduzir o uso de memória, a menos que se excedesse o megabyte. Nos programas para os quais a manipulação de arquivos muito grandes não era crucial,

incentivamos os programadores a ler um arquivo completo na memória e, em seguida, varrer o conteúdo sem se preocuparem com E/S.
Estas decisões permitiram que muitos programas GNU superassem em confiabilidade e em velocidade os seus correspondentes Unix.

Computadores doados

À medida que a reputação do projeto GNU crescia, as pessoas começaram a oferecer máquinas que rodavam Unix como doação para o projeto. Elas eram muito úteis, pois a maneira mais fácil de desenvolver componentes do GNU era fazê-lo em um sistema Unix, substituindo um a um os componentes do sistema. Mas levantavam uma questão ética: se seria correto para nós possuir qualquer cópia do Unix.
O Unix era (e é) software proprietário, e a filosofia do projeto GNU diz que não deveríamos usar software proprietário. Mas, aplicando o mesmo raciocínio que permite concluir que a violência em legítima defesa é justificável, considerei que era legítimo usar um pacote proprietário quando isto fosse crucial para o desenvolvimento de um substituto livre que ajudasse a outros a deixar de usar o pacote proprietário.
Contudo, ainda que fosse um mal justificável, não deixava de ser um mal. Hoje já não possuímos mais cópias do Unix, porque as substituímos por sistemas operacionais livres. Quando não pudemos substituir um sistema operacional de uma máquina por outro livre, substituímos a própria máquina.

A lista de tarefas GNU

Como o projeto GNU prosseguia e eram encontrados ou desenvolvidos um número crescente de componentes do sistema, em certo momento tornou-se útil elaborar uma lista das lacunas remanescentes. Nós a usamos para recrutar desenvolvedores que escrevessem as partes faltantes. A lista se tornou conhecida como a lista de tarefas GNU ("*the GNU task list*"). Nós acrescentamos, além dos componentes Unix que faltavam, vários outros projetos de software útil e de documentação que, segundo pensávamos, deveriam constar de um sistema verdadeiramente completo.
Hoje, já não sobrou praticamente nenhum componente Unix na lista de tarefas GNU -- aqueles trabalhos já foram terminados, sem contar alguns não essenciais. Mas a lista está

cheia de projetos que alguns podem chamar de "aplicações". Qualquer programa que seja atraente para mais que uma estreita categoria de usuários seria algo útil para se incluir em um sistema operacional.
Mesmo jogos estão incluídos na lista de tarefas -- e estiveram desde o princípio. O Unix incluía jogos, logo, naturalmente, o GNU também. Mas, compatibilidade não é problema para jogos, de modo que nós não seguimos a lista de jogos que o Unix tinha. Ao invés disso, listamos um espectro de diferentes tipos de jogos de que os usuários poderiam gostar.

A GPL para Bibliotecas GNU

A biblioteca C do GNU usa um tipo especial de *copyleft* denominado "*GNU Library General Public License*" [*LGPL*, Licença Pública Geral de Biblioteca GNU], que permite que se ligue software proprietário à biblioteca. Por que fazer esta exceção?
Isto não é uma questão de princípios; não há qualquer princípio que diga que produtos de software proprietário têm o direito de incluir o nosso código. (Por que contribuir para um projeto que se sustenta na recusa em compartilhar conosco?) O uso do *LGPL* para a biblioteca C, ou para qualquer outra biblioteca, é uma questão de estratégia.
A biblioteca C realiza um trabalho genérico; qualquer sistema proprietário ou compilador vem com uma biblioteca C. Portanto, tornar a nossa biblioteca disponível apenas para o software livre não traria qualquer vantagem para o software livre -- apenas desestimularia o uso da nossa biblioteca.
Existe um sistema que é uma exceção à regra: no sistema GNU (e isto inclui os sistemas GNU/Linux), a biblioteca C do GNU é a única biblioteca C. De modo que as cláusulas de distribuição da biblioteca C do GNU são as que determinam se é possível ou não compilar um programa proprietário para o sistema GNU. Não há qualquer razão ética para permitir a existência de aplicações proprietárias no sistema GNU, mas, estrategicamente, parece que a proibição ajudaria mais a desencorajar o uso do sistema GNU do que encorajar o desenvolvimento de aplicações livres.
Esta é a razão pela qual o uso da *LibraryGPL* é uma boa estratégia para a biblioteca C. Para outras bibliotecas, a decisão estratégica deve ser considerada caso a caso. Quando uma biblioteca realiza um trabalho especial que pode ajudar na escrita de certo tipo de programas, então, distribuí-la sob a *GPL*, que é permitida somente aos programas livres, é

uma forma de ajudar a outros desenvolvedores de software livre, dando-lhes alguma vantagem frente ao software proprietário.
Considere-se a *GNU Readline*, uma biblioteca desenvolvida para permitir a edição de linha de comando para o *BASH*. A *Readline* se distribui sob a *GNU GPL* comum, e não sob a *LibraryGPL*. Isto provavelmente reduz o quanto a *Readline* é usada, mas não representa uma perda para nós. Enquanto isso, como pelo menos uma aplicação útil se tornou software livre especificamente para poder usar a *Readline*, houve aqui um ganho real para a nossa comunidade.
Os desenvolvedores de software proprietário têm as vantagens que o dinheiro proporciona; os desenvolvedores de software precisam criar vantagens entre si. Tenho a esperança de que algum dia teremos uma grande coleção de bibliotecas cobertas pela *GPL* sem paralelo no software proprietário, proporcionando módulos úteis que sirvam como blocos construtivos no novo software livre, acumulando uma vantagem muito mais importante para o desenvolvimento de software livre subseqüente.

Coçando uma coceira?

Eric Raymond diz que "todo trabalho bem feito de software começa com um desenvolvedor coçando uma coceira pessoal". Talvez isto ocorra algumas vezes, mas várias partes essenciais do software GNU foram desenvolvidas tendo em vista obter um sistema operacional livre completo. São provenientes de uma visão e de um plano, não de um impulso.
Por exemplo, desenvolvemos a biblioteca C do GNU porque um sistema do tipo Unix [*Unix-like*] precisa de uma biblioteca C, o *Bourne Again SHell* (*BASH*) porque um sistema do tipo Unix precisa de um *shell*, e o *tar* [N.T.: *t*ape *ar*chive] do GNU porque um sistema do tipo Unix precisa de um programa *tar*. Isto também é verdade para os meus próprios programas -- o compilador C do GNU, o Emacs do GNU, o *GDB* [N.T.: *GNU debugger*] e o *Make* do GNU.
Alguns programas do GNU foram desenvolvidos para enfrentar ameaças específicas à nossa liberdade. Assim, desenvolvemos o *gzip* para substituir o programa *Compress*, perdido para a nossa comunidade por causa das patentes *LZW*. Financiamos pessoas para desenvolver o *LessTif* e, mais recentemente, iniciamos o *GNOME* e o *Harmony* para tratar

dos problemas causados por certas bibliotecas proprietárias (veja abaixo). Estamos desenvolvendo o *GNU Privacy Guard* para substituir um software popular não livre de criptografia, porque os usuários não deveriam ter que escolher entre privacidade e liberdade.

Certamente, os que escrevem tais programas passaram a ter interesses no trabalho, e muitas características foram acrescentadas por várias pessoas tendo em vista os seus próprios interesses e necessidades. Mas não é este o porquê da existência dos programas.

Desenvolvimentos inesperados

No princípio do projeto GNU, imaginei que desenvolveríamos o sistema completo GNU e o distribuiríamos, então, como um todo. Não foi assim que aconteceu. Como cada componente do sistema GNU era implementado em um sistema Unix, cada componente poderia ser executado em sistemas Unix muito antes que um sistema GNU completo existisse. Alguns daqueles programas se tornaram populares, e os usuários começaram a extendê-los e a transportá-los -- para as diversas versões incompatíveis do Unix e, às vezes, para outros sistemas também.

O processo tornou estes programas muito mais poderosos e atraiu tanto financiamentos como colaboradores para o projeto GNU. Mas, provavelmente, também adiou por vários anos a finalização de um sistema com mínima operacionalidade, já que o tempo dos desenvolvedores do GNU era gasto com a manutenção desses transportes e com a adição de novas características aos componentes existentes, e não com o esforço de escrever sucessivamente cada um dos componentes faltantes.

O GNU Hurd

Em 1990, o sistema GNU estava quase completo; o único componente importante que faltava era o núcleo [*kernel*]. Decidimos implementar o nosso núcleo como uma coleção de processos servidores rodando sobre o *Mach*. O *Mach* é um micro-núcleo desenvolvido na *Carnegie Mellon University* e, depois, na *University of Utah*; o *GNU HURD* é uma coleção de servidores (ou uma "*herd of gnus*", "manada de gnus") que rodam sobre o *Mach*, realizando as diversas tarefas do núcleo Unix. O começo do desenvolvimento foi adiado, na

espera da distribuição do *Mach* como software livre, conforme prometido. Um motivo para a escolha daquele projeto tinha sido evitar o que parecia ser a pior parte do trabalho: depurar um programa de núcleo sem um depurador a nível de código-fonte para fazê-lo. Esta parte do trabalho já havia sido feita, no *Mach*, e pensávamos depurar os servidores *HURD* como programas usuários, com o *GDB*. Mas levou muito tempo para que isto se tornasse possível, e os servidores *multithread* que enviam mensagens uns aos outros se tornaram muito difíceis de depurar. Fazer com que o *HURD* trabalhasse de forma sólida se estendeu por vários anos.

Alix

Originalmente, não se pensava chamar de *HURD* o núcleo GNU. O seu nome original era *Alix* - assim chamado por causa de uma mulher por quem eu estava apaixonado na época. Ela, uma administradora de sistema Unix, havia observado que o seu nome satisfazia um padrão de nomenclatura comum para as versões do sistema Unix; brincando, disse aos seus amigos: "alguém deveria dar o meu nome a um núcleo". Eu não disse nada, mas decidi surpreendê-la com um núcleo chamado Alix.

Não permaneceu assim. Michael Bushnell (agora Thomas), o principal desenvolvedor do núcleo, preferiu o nome *HURD*, redefinindo *Alix* para referir-se a certa parte do núcleo. -- a parte que interceptaria as chamadas de sistema e as trataria mandando mensagens para os servidores *HURD*.

Finalmente, Alix e eu nos separamos, e ela mudou de nome; independentemente, o projeto *HURD* foi alterado, para que a biblioteca C mandasse mensagens diretamente aos servidores, e isto fez com que o componente Alix desaparecesse do projeto.

Mas, antes que tudo isso acontecesse, um amigo dela topou com o nome Alix no código-fonte do *HURD* e lhe contou isso. De modo que o nome cumpriu com o seu objetivo.

Linux e GNU/Linux

O *GNU HURD* não está pronto para o uso em produção. Felizmente, outro núcleo está disponível. Em 1991, Linus Torvalds desenvolveu um núcleo compatível com o Unix e o chamou de Linux. Por volta de 1992, a combinação do Linux com o sistema ainda

incompleto GNU resultou em um sistema operacional livre completo. (Combiná-los foi um trabalho por si só considerável.) De fato, é graças ao Linux que podemos rodar hoje uma versão do sistema GNU.
Chamamos esta versão de sistema de GNU/Linux, para expressar a sua composição como uma combinação do sistema GNU com o Linux como núcleo.

Desafios no nosso futuro

Provamos a nossa capacidade para desenvolver um amplo espectro de software livre. Isto não significa que somos invencíveis ou que nada pode fazer-nos parar. Vários desafios tornam incerto o futuro do software livre; vencê-los exigirá constantes esforços e resistência, às vezes por anos a fio. Isto vai requerer o tipo de determinação que as pessoas mostram quando dão valor à sua liberdade e não permitem que ninguém a tome.
As quatro seções seguintes discutem esses desafios.

Hardware secreto

Os fabricantes de hardware tendem, cada vez mais, a manter em segredo as especificações de hardware. Isto torna difícil escrever programas controladores [*drivers*] livres para que o Linux e o XFree86 possam dar suporte a novos [dispositivos] de hardware. Temos hoje sistemas livres completos, mas não os teremos amanhã se não pudermos dar suporte aos computadores do amanhã.
Existem duas maneiras de enfrentar este problema. Os programadores podem fazer engenharia reversa para entender como dar suporte ao hardware. O resto de nós pode dar preferência ao hardware que seja suportado pelo software livre; à medida que o nosso número aumente, a política de segredo nas especificações terminará por derrotar a si mesma.
Engenharia reversa dá um trabalho imenso; teremos programadores com determinação suficiente para se encarregarem dela? Sim -- se tivermos construído um forte sentimento de que o software livre é uma questão de princípio, e que os programas controladores não livres são intoleráveis. E será que nós, em grande número, gastaríamos dinheiro extra, ou mesmo um pequeno tempo extra para que pudéssemos usar controladores livres? Sim, se a

determinação de ser livre estiver difundida.

Bibliotecas não livres

Uma biblioteca não livre que roda sobre sistemas operacionais livres atua como uma armadilha para os desenvolvedores de software livre. Os elementos atrativos da biblioteca são a isca; se você usa a biblioteca, você cai na armadilha, uma vez que o seu programa não poderá, de forma útil, fazer parte de um sistema operacional livre. (Estritamente falando, poderíamos incluir o seu programa, mas ele não poderia ser *executado* sem a biblioteca faltante.) Pior ainda, se um programa que usa a biblioteca proprietária se torna popular, ele pode atrair outros programadores desavisados em direção à armadilha.

A primeira ocorrência deste problema foi o kit de ferramentas *Motif*, lá pelos anos 80. Embora não existissem sistemas operacionais livres até então, estava claro que tipo de problema o *Motif* lhes ia causar mais tarde. O Projeto GNU respondeu de duas maneiras: solicitando que os projetos individuais de software livre dessem suporte tanto aos dispositivos do kit de ferramentas livre *X* como ao *Motif*, e solicitando que alguém escrevesse um substituto livre para o *Motif*. A tarefa consumiu vários anos; o *LessTif*, desenvolvido pelos *Hungry Programmers* ["Programadores Famintos"], tornou-se poderoso o suficiente para o suporte da maior parte das aplicações *Motif* apenas em 1997.

Entre 1996 e 1998, outra biblioteca não livre de ferramentas GUI ["*Graphical User Interface*", "Interface Gráfica do Usuário"], chamada *Qt*, foi usada em uma coleção substancial de software livre, o *desktop KDE*.

Os sistemas livres GNU/Linux estavam impossibilitados de usar o *KDE*, porque não podíamos utilizar a biblioteca. Contudo, alguns distribuidores comerciais do GNU/Linux, não tão rigorosos ao se vincularem com o software livre, adicionaram o *KDE* aos seus sistemas -- produzindo um sistema com maior capacidade, mas com menos liberdade. O grupo *KDE* incentivava ativamente a mais programadores no uso da *Qt*, e milhões de novos "usuários Linux" jamais foram alertados para a idéia de que havia um problema nisso. A situação parecia sinistra.

A comunidade do software livre respondeu ao problema de duas maneiras: com o *GNOME* e com o *Harmony*.

GNOME, o *GNU Network Object Model Environment* [Ambiente de Rede Modelo de

Objeto GNU] é o projeto de *desktop* do GNU. Começado em 1997, por Miguel de Icaza, e desenvolvido com o apoio de *Red Hat Software*, o *GNOME* passou a oferecer facilidades de *desktop* semelhantes, mas usando exclusivamente software livre. Ele tem vantagens técnicas, também, tal como permitir uma série de linguagens, não apenas o C++. Mas a sua finalidade principal era a liberdade: não exigir o uso de qualquer software não livre. O *Harmony* é uma biblioteca de substituição compatível, projetada para tornar possível a execução do software *KDE* sem usar a *Qt*.

Em novembro de 1998, os desenvolvedores da *Qt* anunciaram uma modificação de licença, que, ao tornar-se efetiva, fará da *Qt* um software livre. Não se pode saber ao certo, mas penso que isto ocorreu, em parte, graças à firme resposta da comunidade ao problema que a *Qt* levantou quando não era livre. (A nova licença é inconveniente e injusta, de modo que ainda é desejável que se evite o uso da *Qt*.)

Como responderemos à próxima biblioteca não livre tentadora? Será que a totalidade da comunidade entenderá a necessidade de permanecer fora da armadilha? Ou será que muitos de nós trocaremos a liberdade pela conveniência, produzindo um problema importante? Nosso futuro depende da nossa filosofia.

Patentes de software

A pior ameaça que enfrentamos provém das patentes de software, que podem colocar certos algoritmos e características fora do alcance do software livre por até vinte anos. As patentes do algoritmo de compressão *LZW* foram requeridas em 1983, e até agora não podemos distribuir software livre que produza GIFs comprimidos de forma adequada. Em 1998, um programa livre que produzia áudio comprimido MP3 foi retirado de distribuição sob a ameaça de um processo de patente.

Existem modos de enfrentar as patentes: podemos procurar por evidências de que uma patente não tem validade, e podemos buscar formas alternativas de realizar um trabalho. Mas cada um destes métodos funciona apenas às vezes: quando ambos falham, uma patente pode forçar todo o software livre a deixar de possuir alguma característica desejada pelos usuários. O que faremos quando isto acontecer?

Aqueles de nós que damos valor ao software livre por amor à liberdade permaneceremos com o software livre, de qualquer modo. Algo faremos para realizar o nosso trabalho sem

as características patenteadas. Mas aqueles que dão valor ao software livre porque esperam que este seja tecnicamente superior, no momento em que as patentes o atrasem, provavelmente irão considerá-lo falho. Por isso, mesmo que seja útil falar da efetividade prática do modelo "catedral" de desenvolvimento e da confiabilidade e do poder de certo software livre, não devemos parar por aí. Devemos falar de liberdade e de princípios.

Documentação livre

A deficiência maior do nosso sistema operacional livre não se encontra no software -- está na falta de bons manuais livres que possamos incluir nos 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, isto é uma grande lacuna. Temos muitas dessas lacunas atualmente.

A documentação livre, assim como o software, é uma questão de liberdade, não de preço. O critério do manual livre é bem parecido ao do software livre: é uma questão de dar aos usuários certas liberdades. A redistribuição (incluindo a venda comercial) deve ser permitida, *on-line* e em papel, de forma que o manual possa acompanhar cada cópia do programa.

A autorização para modificações é igualmente crucial. Como regra geral, não considero essencial que as pessoas tenham autorização para modificar qualquer tipo de artigos ou livros. Por exemplo, não penso que você ou eu estejamos obrigados a permitir que se modifiquem artigos como este aqui, que descreve as nossas ações e as nossas visões. Mas existe um motivo particular pelo qual a liberdade de modificação é crucial para a documentação do software livre. Quando as pessoas exercem os seus direitos de modificar o software, agregando ou alterando as suas características, elas, se forem conscientes, também modificarão o manual -- de modo a poder proporcionar uma documentação precisa e útil com o programa modificado. Um manual que não permite aos programadores serem conscientes e terminarem o trabalho não satisfaz as necessidades da nossa comunidade. Alguns tipos de limite quanto ao modo de se fazerem as modificações não implicam problemas. Por exemplo, o requisito de que se preserve a indicação original de *copyright* do autor, as cláusulas de distribuição, ou a relação de autores, tudo isso está bem. Também não há problema algum em exigir que as versões modificadas incluam a observação de que elas

foram modificadas, ou mesmo que existam seções inteiras que não possam ser apagadas ou alteradas, desde que tais seções versem sobre assuntos não técnicos. Estes tipos de restrições não são um problema porque não impedem o programador consciencioso de adaptar o manual para ajustá-lo ao programa modificado. Em outras palavras, eles não impedem que a comunidade do software livre faça uso pleno do manual. Contudo, deve ser possível modificar todo o conteúdo *técnico* do manual e distribuir o resultado, em seguida, em todas as mídias usuais, através de todos os canais usuais; senão, as restrições sim obstruem a comunidade, o manual não é livre, e nós necessitamos de outro manual.

Terão os desenvolvedores de software livre a consciência e a determinação para produzir um espectro completo de manuais livres? Uma vez mais, o nosso futuro depende da nossa filosofia.

Devemos falar de libertade

Há estimativas hoje de que existem uns dez milhões de usuários de sistemas GNU/Linux, tais como o *Debian* GNU/Linux e o *Red Hat* Linux. O software livre desenvolveu tais vantagens práticas que os usuários estão migrando para ele por razões puramente práticas. As boas conseqüências disso são evidentes: mais interesse em desenvolver software livre, mais clientes para os negócios do software livre e mais capacidade para incentivar as companhias a desenvolverem software livre comercial ao invés de produtos de software proprietário.

Mas o interesse no software está crescendo mais rapidamente do que a consciência a respeito da filosofia na qual ele se baseia, e isso traz problemas. A nossa capacidade de enfrentar os desafios e ameaças acima descritas depende da vontade de permanecer firmemente do lado da liberdade. Para garantir que a nossa comunidade tenha esta vontade, precisamos difundir a idéia para os novos usuários à medida que eles chegam à comunidade.

Mas estamos fracassando nisso: os esforços para atrair novos usuários para a nossa comunidade superam de muito os esforços para ensinar-lhes a cidadania da nossa comunidade. Precisamos fazer ambas as coisas, e precisamos manter os dois esforços

equilibrados.

"Open Source"

Em 1998, tornou-se mais difícil ensinar aos novos usuários sobre a liberdade, quando uma parte da comunidade decidiu parar de usar o termo "software livre" e, ao invés disso, dizer "*open source software*" [software de código-fonte aberto].
Alguns daqueles que preferiram esta expressão desejavam evitar a confusão de "livre" com "grátis" [N.T.: em inglês, *free* = grátis ou livre] -- um objetivo válido. Outros, contudo, desejavam colocar de lado o espírito do princípio que motivou o movimento do software livre e o projeto GNU e, ao invés disso, atrair a executivos e usuários comerciais, muitos dos quais abraçam uma ideologia que coloca o lucro acima da liberdade, acima da comunidade, acima dos princípios.
Assim, a retórica do "*open source*" focaliza o potencial de criação de software poderoso de alta qualidade, mas evita as idéias de liberdade, de comunidade e de princípios.
As revistas sobre o "Linux" são um exemplo claro disso -- estão cheias de propagandas de software proprietário que funcionam com o GNU/Linux. Quando os próximos *Motif* ou *Qt* aparecer, será que estas revistas alertarão aos programadores para que permaneçam longe deles, ou farão a sua propaganda?
O apoio comercial pode contribuir para a comunidade de muitas formas; todo o resto permanecendo igual, é útil. Mas obter o seu apoio falando menos ainda de liberdade e de princípios pode ser desastroso; faz com que o desequilíbrio prévio entre a difusão e a educação cívica se torne ainda pior.
"Software livre" e "*open source*" descrevem, mais ou menos, a mesma categoria de software, mas dizem coisas diferentes sobre o software e sobre os valores. O Projeto GNU continua a usar a expressão "software livre", para exprimir a idéia de que a liberdade, e não somente a tecnologia, é o que importa.

Tente!

A filosofia de Yoda [*sic*] ("não existe 'tentar'") soa bem, mas não funciona comigo. Fiz a maior parte do meu trabalho angustiado por saber se poderia realizá-lo e inseguro sobre se,

alcançando a meta, isto seria suficiente. Mas, de qualquer modo, tentei, porque não havia ninguém mais entre o inimigo e a minha cidade. Para a minha própria surpresa, tive êxito algumas vezes.
Falhei algumas vezes; algumas das minhas cidades caíram. Então, encontrava outra cidade ameaçada e me preparava para outra batalha. Ao longo do tempo, aprendi a procurar por ameaças e a colocar-me entre elas e as minhas cidades, chamando outros *hackers* para se juntarem a mim.
Atualmente, freqüentemente não estou sozinho. É um consolo e um prazer quando vejo um regimento de *hackers* cavando para manter a trincheira, e percebo que esta cidade sobreviverá -- por enquanto. Mas os perigos são maiores a cada ano que passa, e agora a Microsoft colocou a nossa comunidade explicitamente na sua mira. Não podemos considerar o futuro da liberdade garantido. Não o considere garantido! Se você deseja conservar a sua liberdade, deve estar preparado para defendê-la.

Notas:

[1]. O uso da palavra "*hacker*" para descrever alguém que "quebra a segurança" é uma confusão proveniente da mídia. Nós, os *hackers*, nos negamos a reconhecer tal acepção e continuamos a usar a palavra com o sentido de "alguém que tem paixão por programar e que adora ser engenhoso ao fazê-lo".
The use of "hacker" to mean "security breaker" is a confusion on the part of the mass media. We hackers refuse to recognize that meaning, and continue using the word to mean, "Someone who loves to program and enjoys being clever about it."

[2]. Como ateu que sou, não sigo nenhum líder religioso, mas às vezes acho que admiro alguma coisa dita por algum deles.

[3]. Em 1984 ou 1985, Don Hopkins (uma pessoa muito imaginativa) enviou-me uma carta. No envelope havia várias expressões divertidas, entre as quais a seguinte: "*Copyleft -- all rights reversed*" [todos os direitos "invertidos"]. Usei a palavra "*copyleft*" para denominar o conceito de distibuição que eu estava desenvolvendo na época. [N.T.: *left* = esquerda;

deixado (do verbo *leave*, deixar; *with your leave* = com a sua licença)]

[4]. "*Bourne Again SHell*" é uma brincadeira com o nome "*Bourne Shell*", que era o shell usual do Unix. [N.T.: *born again* = nascer de novo; *Bourne* é o autor do shell usual do Unix]

Link para outros artigos e ensaios relacionados: http://www.gnu.org/philosophy e http://lpf.ai.mit.edu/Legal/legal.html

Sobre o artigo / About the Paper:
Publicado originalmente no livro «Open Sources». Tradução (LCBP, para a revista *DataGramaZero*) da versão atualizada em inglês disponível em http://www.gnu.org/gnu/thegnuproject.html
Por favor, envie as suas perguntas (em inglês) sobre FSF & GNU a *gnu@gnu.org*. Há outras formas de contactar a FSF.
Por favor, envie os seus comentários (em inglês) sobre as páginas do GNU Project a *webmasters@www.gnu.org*, envie outras perguntas (em inglês) a *gnu@gnu.org*.

****Sobre o autor / About the Author:**

Richard Stallman
rms@gnu.org - http://www.gnu.org/people/rms.html

Richard Stallman é o fundador do Projeto GNU, lançado em 1984 para desenvolver o sistema operacional livre GNU (um acrônimo para "GNU's Not Unix - Gnu Não é Unix") e, assim, restituir aos usuários de computadores a liberdade que a maior parte deles perdeu. GNU é software livre: todo mundo tem a liberdade de copiá-lo e redistribuí-lo, assim como modificá-lo, em grande ou pequena escala. Hoje, as variantes Linux do sistema GNU, baseadas no núcleo (kernel) Linux desenvolvido por Linus Torvalds, são usadas amplamente. Estima-se, hoje, em mais de 10 milhões o número de usuários de sistemas GNU/Linux. Richard Stallman é o autor principal do Compilador C GNU ("GNU C Compiler"), que suporta atualmente mais de 30 arquiteturas diferentes e sete linguagens de programação, do depurador " GNU symbolic debugger (GDB)", do "GNU Emacs", e outros. Recebeu o Prêmio "Grace Hopper" da Association for Computing Machinery (ACM), em 1991, pelo desenvolvimento do primeiro editor "Emacs", nos anos 70. Em 1990, foi premiado com uma bolsa da MacArthur Foundation e, em 1996, como Doutor Honoris Causa pelo Instituto Real de Tecnologia da Suécia. Em 1998, recebeu o Prêmio "Electronic Frontier Foundation's Pioneer", juntamente com Linus Torvalds. Recebeu o Prêmio Yuri Rubinski em 1999.

Disponível em: http://www.dgz.org.br/fev00/Art_04.htm
Acesso em: 17 de setembro de 2007