Projeto GNU

por Richard Stalmann

(publicado originalmente no livro ``*open sources*'')

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 *hackers* [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*'', como em ``*freedom*'' --``*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 Environmen* (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''

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. È 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:

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. Atualizado em 20 /mar/2000

Tradução: Alexandre J. Thomé (Brasil) - alexandre.thome@intra.procergs.com.br - 12/abr/2000

Revisão Geral : Eliane M. de Azevedo (Brasil) - eliane.azevedo@intra.procergs.com.br - 13/abr/2000

Revisão Técnica: Ronaldo Cardozo Lages (Brasil) - rclages@intra.procergs.com.br - 14/abr/2000

Publicado na Internet no Site do Deputado Estadual Elvino Bohn Gass-PT/RS: http://www.al.rs.gov.br/deputados/bohn_gass

Adaptação para formato LaTeX: Elgio Schlemer - schlemer5@terra.com.br - 05/jul/2000

---

[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.

This document was translated from $LaTeX$ by HEVEA.